CHAPTER

6

INTRODUCTION TO HARD DRIVES

In this chapter, you will learn:

- How data is stored on a hard drive
- How to use DOS and Windows commands to manage data on a hard drive
- How to identify the various types of hard drives and understand the advantages of each
- How to manage a hard drive to optimize its performance

In this chapter, the discussion of floppy disks from Chapter 5 is extended to hard drives. You can apply much of what you learned about the way floppy disks hold and manage data to hard drives. Just like a floppy disk, a hard drive has a file allocation table (FAT) and root directory, and stores data on tracks that are divided into sectors, each of which contains 512 bytes. In addition, many Windows and DOS commands that manage files on a floppy disk are used on hard drives as well. However, because of its size and versatility, a hard drive has more complicated methods of formatting and organizing data.

This chapter introduces hard drive basics and explains how to manage a healthy, previously installed hard drive. Chapter 7 continues our discussion of hard drives, examining installation, troubleshooting, and data recovery.

HARD DRIVE TECHNOLOGY

Hard drives used in today's microcomputers have their origin in the hard drives of early mainframe computers of the 1970s. These drives consisted of large platters or disks that were much larger and thicker than phonograph records. Several platters were stacked together with enough room to allow read/write heads to move back and forth between the platters. All heads moved in unison while the platters spun at a high speed. Applications programmers of the 1970s were responsible for how and where data was written to the platters. They wrote their programs so that data was spaced evenly over the disks, meaning the heads moved as little as possible while reading or writing a file. They could judge their success by standing over the clear cover of the hard drive and watching the heads. If they had programmed well, the heads moved smoothly over the platters; if not, they thrashed back and forth as data was processed. In today's systems, things are much more complicated. Several layers of software separate the data stored on a hard drive or floppy disk and the applications that read data from or write data to the disk. Learning about these layers, how they relate to one another, and how they manage the hard drive are the focus of this chapter.

A+CORE 1.1

Hard drive structure and function has not changed, however. Modern hard drives have two or more platters that are stacked together and spin in unison. Read/write heads are controlled by an actuator and move in unison across the disk surfaces as the disks rotate on a spindle (see Figure 6-1). PCs can use several types of hard drives, all using a magnetic medium; the data on all these drives is stored in tracks and sectors, and data files are addressed in clusters made up of one or more sectors.

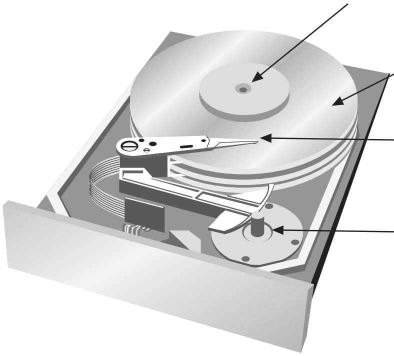

Figure 6-1 Inside a hard drive case

Figure 6-2 shows a hard drive with four platters. All eight sides of these four platters are used to store data, although on some hard drives the top side of the first platter just holds information used to track the data and manage the disk. As with floppy drives, each side or surface of one hard drive platter is called a **head**. (Don't confuse this with the read/write mechanism that moves across a platter, which is called a read/write head.) The drive in Figure 6-2 has eight heads. Also as with floppy drives, each head is divided into tracks and sectors. The eight tracks shown in Figure 6-2, all of which are the same distance from the

A+ CORE 1.1

center of the platters, together make up one cylinder. If a disk has 300 tracks per head, then it also has that same number of cylinders.

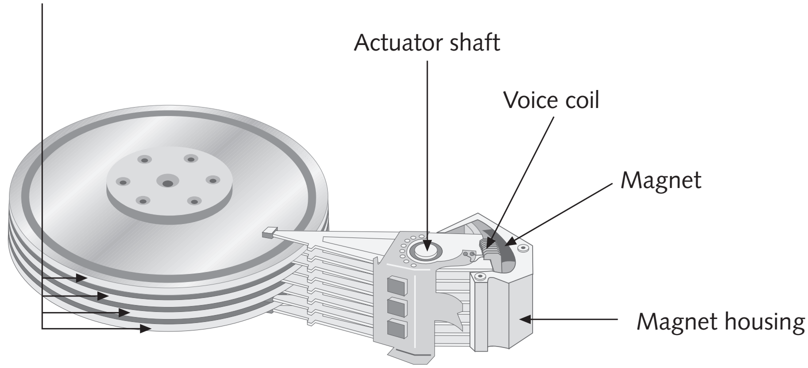

Figure 6-2 A hard drive with four platters

Just as with floppy disks, data is written to a hard drive beginning at the outermost track. The entire first cylinder is filled before the read/write heads move inward and begin filling the second cylinder. For older hard drives, even though tracks closer to the center of a platter are smaller, these tracks must store the same amount of data as the larger tracks toward the outside of a platter. As the heads move toward the center of the drive and the tracks become smaller, the read/write heads must adjust the way they write data so that sectors store a consistent number of bytes, even if they are different physical sizes. Two methods can be used to adjust for the smaller tracks: write precompensation and reduced write current.

Write precompensation speeds up the writing of data to the drive as the tracks become smaller near the center of the platters. If a hard drive uses write precompensation, it indicates at what track or cylinder the precompensation begins. Appendix B shows that some drives don't use this method. Some tables list the write precompensation as the total number of cylinders for the drive type. Interpret this to mean that precompensation is not used.

Reduced write current means just what it says. At a cylinder near the center of the platter, the read/write heads reduce the current used to place magnetized spots on the disk, because the spots are getting closer and closer together. Reduced write current is not as common as write precompensation.

IDE Technology

A+ CORE 1.1

Almost all hard drives on the market today use **IDE (Integrated Device Electronics** formerly Integrated Drive Electronics) standards, but it was not always so. Older hard drive technologies, a few of which are still in use, include MFM, RLL, and ESDI. A variation of IDE technology is Enhanced IDE (EIDE). SCSI is one technology that manages the interface between the hard drive and the system bus.

The following sections discuss how IDE and SCSI work; MFM and RLL drives are covered for historical reasons only, to help you understand the basics behind today's drive technology.

As described earlier, a hard drive consists of two or more platters spinning inside a protective housing with read/write heads that move back and forth across the platters. The drive fits into a bay inside the computer case and is securely attached with supports or braces and screws. This helps prevent the drive from being jarred while the disk is spinning and the heads are very close to the disk surfaces.

A+ CORE 1.2

A hard drive requires a controller board filled with ROM programming to instruct the read/write heads how, where, and when to move across the platters and write and read data. In IDE and SCSI drives, a controller is mounted on a circuit board on the drive housing and is an integral part of it. (Hence the term Integrated Device Electronics, or IDE.) Older RLL and MFM drives had the controller board as a separate, large expansion card connected to the drive with two cables. Today, the controller of an IDE drive usually connects to the system board by way of a data cable from the drive to an IDE connection directly on the system board. Older system boards did not have an IDE connection, so a small **adapter card** served as a simple pass-through from the drive to the system board. The data cable attached to the drive and the adapter card, which was inserted in an ISA slot on the system board. Sometimes an adapter card connects a hard drive to a system board to compensate for the system board BIOS not supporting a large-capacity drive. The adapter card contains the necessary BIOS to support the drive in the place of the system BIOS.

Figure 6-3 shows a hardware subsystem including an IDE hard drive and its connection to a system board. In addition to the connection for the 40-pin data cable, the hard drive has a connection for the power cord from the power supply.

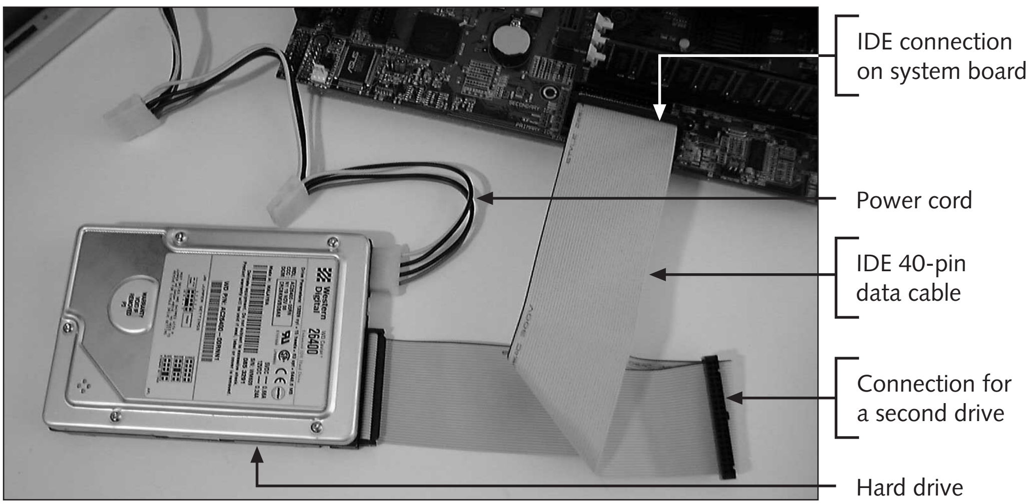

Figure 6-3 A PC's hard drive subsystem

Almost all drives on the market today are IDE technology. Although IDE technology is an innovative improvement over MFM and RLL type hard drives, it also introduces some new limitations. To understand how IDE technology differs from other drive technologies, examine the details of how drives are low-level formatted.

Tracks and Sectors on an IDE Drive

The MFM and RLL technologies use either 17 or 26 sectors per track over the entire drive platter (see Figure 6-4). The larger tracks near the outside of the platter contain the same number of bytes as the smaller tracks near the center of the platter. This arrangement makes the formatting of a drive and later accessing data simpler, but it wastes drive space. The number of bytes that a track can hold is determined by the centermost track, and all other tracks are forced to follow this restriction.

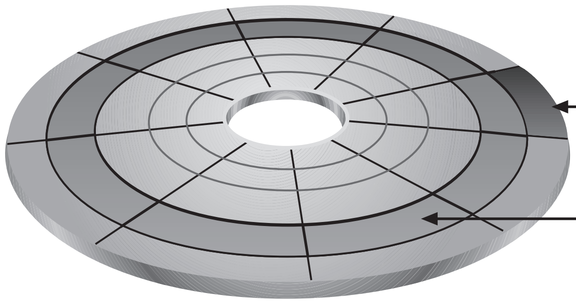

Figure 6-4 Floppy drives and older hard drives use a constant number of sectors per track

One major improvement with IDE technology is that the formatting of IDE drives eliminates this restriction. The number of sectors per track on an IDE drive is not the same throughout the platter. In this new formatting system, called **zone bit recording** (Figure 6-5), tracks near the center have the smallest number of sectors per track, and the number of sectors increases as the tracks get larger. In other words, each track on an IDE drive is designed to have the optimum number of sectors appropriate to the size of the track. What makes this arrangement possible, however, is one fact that has not changed: every sector on the drive still has 512 bytes. If it weren't for this consistency, the OS would have a difficult time indeed communicating with this drive!

Because each track can have a different number of sectors, the OS cannot communicate with an IDE drive by contacting the hard drive controller BIOS and using sector and track coordinates, as it does with floppy disks and older hard drives. Newer, more sophisticated methods must be used that are discussed later in the chapter.

Formatting a Hard Drive

A+ OS 1.3

Recall from Chapter 5 that, when the OS formats a floppy disk, it writes sector and track markings on the disk. With IDE drives, since the track and sector markings no longer follow a simple pattern, they are written on the hard drive at the factory. This process is called

A+ OS 1.3 **low-level formatting**. The OS still executes the remainder of the format process (creating a boot sector, FAT, and root directory), which is called the **high-level format** or **OS format**.

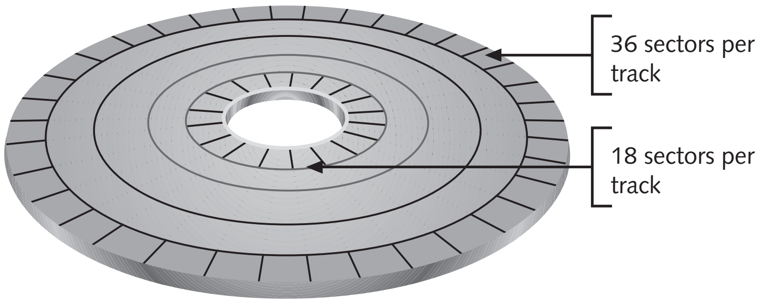

Figure 6-5 Zone bit recording can have more sectors per track as the tracks get larger

With older RLL and MFM technology, the system BIOS, OS, or utility software such as Norton Utilities and SpinRite could perform the low-level format; even now a low-level format routine is still part of standard system BIOS. For an IDE drive, however, using system BIOS or standard utility software to low-level format the drive would be a catastrophe. Formatting an IDE drive in this way could permanently destroy the drive unless the drive controller BIOS were smart enough to ignore the command.

Because of the unique way that an IDE drive is formatted and data is accessed, you must use a controller specific to the IDE drive. IDE drives thus have their controller built directly on top or on the bottom of the drive housing. The controller and the drive are permanently attached to one another.

Because IDE drives are low-level formatted by the manufacturer, they cannot be low-level formatted as part of preventive maintenance as older drives can be. The track and sector markings on the drive created at the factory are normally expected to last for the life of the drive. For this reason IDE drives are often referred to as disposable drives. When the track and sector markings fade, as they eventually do, you just throw the drive away and buy a new one!

A+ OS 3.2 However, improvements for formatting the IDE drive are becoming more commonplace. Some better-known IDE drive manufacturers are offering a low-level format program specific to their drives. If an IDE drive continues to give "Bad Sector or Sector Not Found" errors or even becomes totally unusable, ask the manufacturer for a program to perform a low-level format of the drive. Sometimes these programs are only distributed by the manufacturer to dealers, resellers, or certified service centers.

> TIP: If an IDE drive creates problems, try using the manufacturer's low-level format software to re-create track and sector markings on the drive.

It's risky to low-level format an IDE drive using a format program other than one provided by the manufacturer, although some have tried and succeeded. Probably more drives have been permanently destroyed than saved by taking this risk, however. IDE drives last several

years without a refresher low-level format. By that time, you're probably ready to upgrade to a larger drive anyway.

A+CORE 4.3

System-Board Support for IDE Drives

When IDE drives first entered the marketplace, an adapter card in an expansion slot connected the drive to the system board. Nowadays, most system boards support IDE by providing one or two IDE connections directly on the system board. To alert a user when a hard disk is being accessed, a system board has a two-pin connection that connects to an LED on the front of the computer case. A lit LED means a hard drive is being accessed.

In summary, an IDE drive has its controller mounted directly on top of the drive. The OS does not communicate directly with the IDE drive, as is the case with drives that use older technology. Instead, the OS passes its requests to the drive controller, which is responsible for keeping up with where and how data is stored on the drive. As far as the OS is concerned, an IDE drive is simply a very long list of logical sectors, each 512 bytes long. The OS doesn't care where on the drive these sectors are located, since that information is maintained by the controller. Setup for IDE drives is very simple; the most important fact that setup and the OS need to know is how many sectors there are on the drive. It is important not to overestimate the number of sectors. You don't want the OS requesting use of a sector that does not exist. However, you can tell setup that you have fewer sectors than are present. If you do, some sectors will remain unused.

Enhanced IDE (EIDE) Drives

Early IDE drives followed the IDE/ATA (Integrated Device Electronics AT Attachment) standard[1] developed by ANSI. This standard is sometimes referred to as the IDE standard or the ATA standard. It involves how the drive interfaces with BIOS and software more than it does the actual drive technology. Using this ATA standard interface, drives were limited to 528 MB and could have no more than 1,024 cylinders. There could be no more than two hard drives on the same interface. This standard applied only to hard drives and did not include CD-ROM drives, tape drives, and so on. It has been improved several times as drive technology and methods of interface have improved.

Enhanced IDE (**EIDE**) drives support these newer, faster standards. The first standard supported by EIDE was ATA-2. This new standard allowed for up to four IDE devices on the same PC. These IDE devices could be hard drives, CD-ROM drives, or tape drives as well as other IDE devices. CD-ROM drives use the ATAPI standard. As standards were developed, different hard drive manufacturers adopted different names for them, which can be confusing. Standards today specify data transfer speed more than any other single factor. When selecting a hard drive standard, select the fastest standard you can, but keep in mind that the operating system, BIOS on the system board, and the hard drive must all support this standard. If one of three does not support the standard, the other two will probably revert to using a slower standard that all three can use.

[1]AT originally stood for advanced technology and refers to one of the early IBM PCs of the 1980s. The XT came before the AT PC. Sometimes you still hear the term XT/AT compatible or AT standard, which means the technology follows the standard established by this early AT PC.

A+CORE 4.3

Table 6-1 lists the different ANSI standards for hard drives.

Table 6-1 Summary of ANSI interface standards for hard drives

Standard (May Have More Than One Name)	Speed	Description
IDE/ATA ATA	Speeds range from 2.1 MB/sec to 8.3 MB/sec	The first ANSI hard drive standard for IDE hard drives. Limited to no more than 528 MB. Supports PIO and DMA transfer modes.
ATA-2 Fast ATA	Speeds up to 16.6 MB/sec	Breaks the 528-MB barrier. Allows up to 4 IDE devices. Supports PIO and DMA transfer modes discussed later in the chapter.
ATA-3	Little speed increase	Improved version of ATA-2
Ultra ATA Fast ATA-2 Ultra DMA DMA/33	Speeds up to 33.3 MB/sec	Defined a new DMA mode, but only supports slower PIO modes
Ultra ATA/66 Ultra DMA/66	Speeds up to 66.6 MB/sec	Uses a special 40-pin cable that provides additional ground lines on the cable to improve signal integrity

SCSI Technology

SCSI (pronounced "scuzzy") stands for **Small Computer Systems Interface** and is a standard for communication between a subsystem of peripheral devices and the system bus. The SCSI bus is a closed system that can contain, and be used by, up to seven or 15 devices, depending on the SCSI standard. The gateway from this bus to the system bus is an adapter card inserted into an expansion slot on the system board. The adapter card, called the **host adapter**, is responsible for managing all the devices on the SCSI bus. When one of these devices must communicate with the system bus, the data passes through the host adapter (see Figure 6-6).

The host adapter keeps up with the interchange between the devices on the SCSI bus and the system bus. SCSI technology has the added advantage of letting two devices on the SCSI bus pass data between them without going through the CPU. This method of data transmission provides a convenient way to back up a SCSI hard drive to a tape drive on the same host adapter without involving the CPU.

A+CORE 1.6

The maximum number of devices the SCSI bus can support depends on the type of SCSI being used. Some SCSI buses can link up to seven, others up to 15 devices. Each device on the bus is assigned a number from zero to seven called the **SCSI ID**, using DIP switches, dials on the device, or software settings. The host adapter is assigned a number larger than all other devices, either 7 or 15. Cables connect the devices physically in a straight chain. The devices can be either internal or external, and the host adapter can be at either end of the chain or somewhere in the middle. The SCSI ID identifies the physical device, which can

A+ CORE 1.6

have several logical devices embedded in it. For example, a CD-ROM changer or jukebox device might have seven CD trays. Each tray is considered a logical device and is assigned a **Logical Unit Number (LUN)** to identify it such as 1 through 7 or 0 through 6. The ID and LUN are written as two numbers separated by a colon. For instance, if the SCSI ID is 5, the fourth tray in the jukebox is device 5:4.

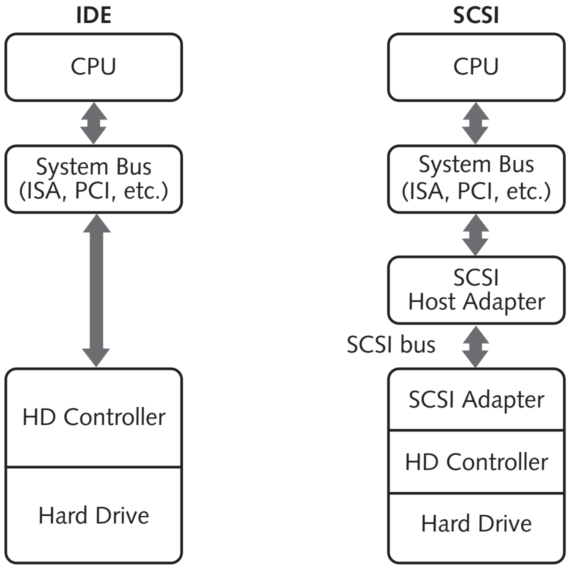

Figure 6-6 SCSI hard drives communicate with the CPU through the SCSI host adapter, but IDE drives communicate directly on the system bus

A SCSI device such as a hard drive, tape drive, or CD-ROM drive, interfaces with the host adapter rather than directly with the CPU. The technology of a SCSI device can be the same as the technology of a similar device that is not SCSI, with the added ability to use the SCSI bus and communicate with the host adapter. A device is a SCSI device not because of its technology, but because of the bus it uses.

Just as with IDE drives, a SCSI drive has its controller mounted inside the drive housing and can have a variable number of sectors per track. Low-level formatting a SCSI drive is sometimes possible and even recommended by the manufacturer because the SCSI controller is likely to contain the firmware to do the job. See the drive's documentation. In theory, a SCSI drive can simply be an IDE drive with one more chip on the controller card in the drive housing and a different kind of data connection designed to fit the SCSI standard. In practice, however, SCSI drives are of higher quality, having higher rotational speeds and lower seek times than IDE drives. The SCSI chip within the drive housing that controls data transfer over the SCSI bus is called the **SCSI bus adapter chip (SBAC)**.

Figure 6-6 illustrates this concept of SCSI as a bus. On the left, the CPU communicates with the hard drive controller, which is contained in the hard drive case, through the system bus. On the right, the CPU communicates over the system bus to the SCSI host adapter, which

communicates over the SCSI bus to the SCSI adapter in the hard drive case. The SCSI adapter communicates with the hard drive controller, which, in turn, communicates with the hard drive.

Sometimes a SCSI device is designed to be the only SCSI device on the SCSI bus and so has some of the host adapter logic built into the device. These devices are called **embedded SCSI** devices. The host adapter for the device is usually bundled with it and is much smaller and simpler than normal host adapters. Other SCSI devices on this computer must be separate from this SCSI bus system. Embedded SCSI devices often don't conform to standard SCSI specifications, because they do not accommodate any other SCSI device.

A+ CORE 1.6

To reduce the amount of electrical "noise," or interference, on a SCSI cable, each end of the SCSI chain has a **terminating resistor**. The terminating resistor can be a hardware device plugged into the last device on each end of the chain, or the chain can have software-controlled termination resistance, which makes installation simpler.

A+ CORE 1.6, 4.3

Differing SCSI Standards Just as with IDE/ATA standards, SCSI standards have improved over the years and use different names. The two general categories of all SCSI standards used on PCs have to do with the number of bits that travel on the SCSI bus, either 8 bits (narrow SCSI) or 16 bits (wide SCSI). In almost every case, if the SCSI standard is 16 bits, then the word "wide" is in the name for the standard. In most cases, the word "narrow" is not mentioned in names for 8-bit standards. Narrow SCSI uses a 50-pin cable, and wide SCSI uses a 68-pin cable.

A SCSI cable can be built in two different ways, depending on the method by which the electrical signal travels on the cable: single-ended and differential. A single-ended cable is less expensive than a differential cable, but the maximum cable length cannot be as long because data integrity is not as great. Cables for both narrow SCSI and wide SCSI can be either single-ended or differential. Single-ended cables and differential cables look the same, so you must make sure that you are using the correct cable. Single-ended cable is more popular than differential because it is less expensive.

Table 6-2 summarizes the different SCSI standards, including the three major standards: SCSI-1, SCSI-2 and SCSI-3 that are more commonly known as Regular SCSI, Fast SCSI, and Ultra SCSI, respectively. Other names used in the industry for these standards are also listed in the table. Both Fast SCSI and Ultra SCSI have narrow and wide versions.

Because there are several variations of SCSI, when you buy a new SCSI device, you must be sure that it is compatible with the SCSI bus you already have. All SCSI standards are not backward compatible with earlier SCSI standards. If the new SCSI device is not compatible, you cannot use the same SCSI bus, and you must buy a new host adapter to build a second SCSI bus system, increasing the overall cost of adding the new device.

The wide SCSI specification allows for a data path of 32 bits, although this has not been implemented in PCs. When you see a SCSI device referred to as wide, you can assume 16 bits.

Beginning with Ultra SCSI (SCSI-3), the SCSI standard supports **SCSI configuration automatically** (**SCAM**), which follows the Plug and Play standard. SCAM makes installing SCSI devices easier, if the device is SCAM-compatible.

A+CORE 1.6, 4.3

Table 6-2 Summary of SCSI standards

Names for the SCSI Interface Standard	Bus width Narrow = 8 bits Wide = 16 bits	Transfer ZRate (MB/sec)	Maximum Length of Single-ended Cable (meters)	Maximum Length of Differential Cable (meters)	Maximum Number of Devices
SCSI-1 (Regular SCSI)[1]	Narrow	5	6	25	8
SCSI-2 (**Fast SCSI** or Fast Narrow)	Narrow	10	3	25	8
Fast Wide SCSI (Wide SCSI)	Wide	20	3	25	16
SCSI-3 (**Ultra SCSI** or Ultra Narrow or Fast-20 SCSI	Narrow	20	1.5	25	8
Wide Ultra SCSI (Fast Wide 20)	Wide	40	1.5	25	16
Ultra2 SCSI	Narrow	40		12 LVD[2]	8
Wide Ultra2 SCSI	Wide	80			16
Ultra3 SCSI	Narrow	80		12 LVD[2]	8
Wide Ultra3 SCSI (Ultra 160 SCSI)	Wide	160		12 LVD[2]	16

[1]Bold indicates most common name
[2]LVD: Low voltage differential cable allows for lengths up to 12 meters

A+CORE 1.6

A sample configuration of a SCSI subsystem is shown in Figure 6-7. Because Ultra SCSI is backward compatible with SCSI-1 and SCSI-2, all three can coexist on the same SCSI bus. Note that the only connection this subsystem has to the overall computer system and the CPU is through the host adapter. You can see from the diagram why some people compare a SCSI system to a miniature LAN inside a computer.

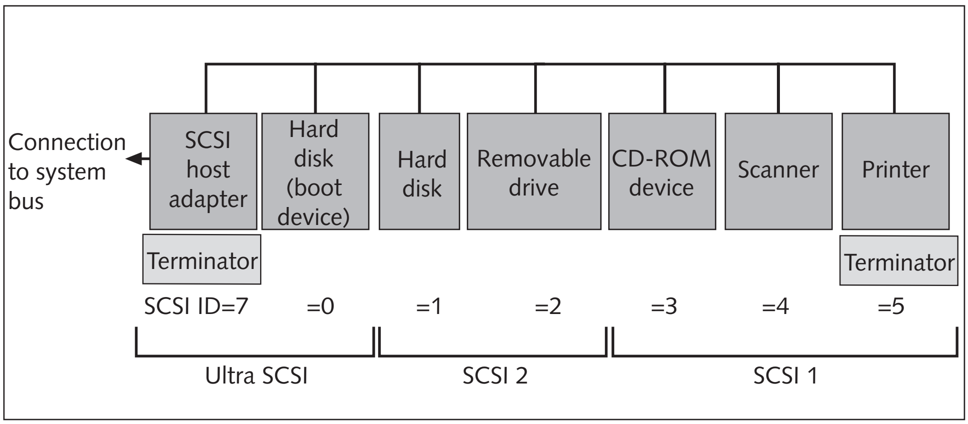

Figure 6-7 Sample SCSI subsystem configuration

Other Variations of SCSI Hardware and Software

In addition to the differences in SCSI standards, other components of SCSI vary. Besides the cables already discussed above, there are also variations in termination, device drivers, and host adapters.

A+ CORE 1.6

Termination **Termination** prevents an echo effect from electrical noise and reflected data at the end of the SCSI daisy chain, which can cause interference with the data transmission. There are several ways to terminate power:

- The host adapter can have a switch setting that activates or deactivates a terminating resistor on the card, depending on whether or not the adapter is at one end of the chain.
- A device can have either a single SCSI connection requiring that the device is placed at the end of the chain, or the device can have two connections. When a device has two connections, the second connection can be used to connect another device or to terminate the chain by placing an external terminator on the connection. This external terminator serves as the terminating resistor (see Figure 6-8).

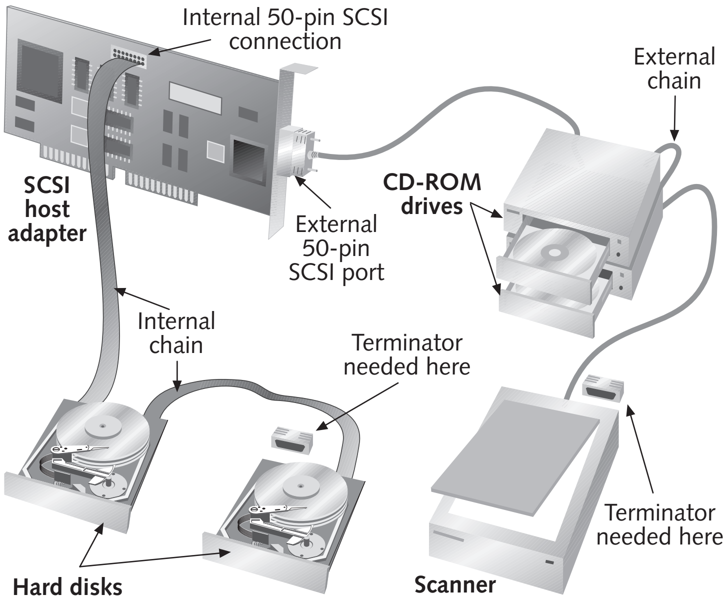

Figure 6-8 SCSI subsystem showing terminators at each end of the SCSI chain

- The device at the end of the chain can also be terminated by a resistor that is physically mounted on the device in a specially designated socket.
- Some devices have built-in terminators that you can turn on or off with a jumper setting on the device.
- Termination can be controlled by software.

There are several types of terminators: passive terminators, active terminators, and forced perfect terminators. Forced perfect terminators are more expensive and more reliable than the other two are.

A+CORE 1.6

When buying terminating resistor hardware and cables for a SCSI bus (see Figure 6-9) get high-quality products even if they cost a little more. The added reliability and enhanced data integrity are worth the extra money.

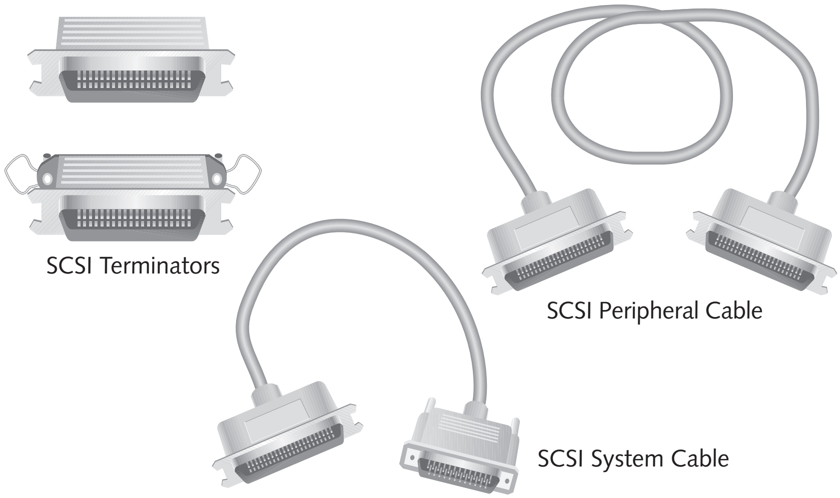

Figure 6-9 SCSI cables and terminators

SCSI Device Drivers SCSI device drivers are needed to enable DOS or another OS to communicate with a host adapter. Although many drivers are available, it is best to use the drivers recommended by or provided by the host adapter vendor. Two popular drivers are the **Advanced SCSI Programming Interface** (**ASPI**) and **Common Access Method** (**CAM**). ASPI is probably the more popular of the two.

No SCSI device drivers are included with Window 3.1 or DOS. SCSI hardware manufacturers write their own drivers and include them with their devices. A SCSI driver is loaded in the CONFIG.SYS file just as other drivers are. Its DEVICE= command must appear in the CONFIG.SYS file before any device drivers for a SCSI device. For example, if you have a SCSI CD-ROM drive whose device driver must be installed in CONFIG.SYS, you must place the command for this driver after the one that installs the host adapter driver. The installation instructions for the CD-ROM drive offer similar instructions and give you the specific command. Windows 9x, Windows NT, and Windows 2000 all have built-in support for SCSI devices.

A+CORE 1.6

Many computers have some SCSI interface software in their system BIOS, enough, in fact, to allow a SCSI hard drive to be the boot device of the system. The system BIOS can access the SCSI drive, execute the load program in the drive's master boot record, and load the SCSI device drivers stored on the hard drive into memory. If a system has two hard drives, one being an IDE and one a SCSI, the IDE drive must be the boot device unless system BIOS can support booting from a SCSI drive even when an IDE drive is present. This is because the system-board BIOS takes precedence over the BIOS on the SCSI host adapter.

Host Adapter Issues An important issue when you install a SCSI bus system for the first time is the sophistication of the host adapter. More expensive host adapters are often easier to install because the installation software does more of the work for the installer and offers more help than does less expensive adapter software. When buying a host adapter, compare the installation procedures for different adapters, and also look for options, such as a built-in disk drive controller, software-controlled termination (eliminating the need for hardware terminating resistors), and configuration BIOS built into the adapter's ROM.

A+CORE 1.3

A SCSI host adapter controller has BIOS that loads into memory addresses on the PC and controls the operation of the SCSI bus. This SCSI controller uses a DMA channel, IRQ, and I/O addresses. You must install a SCSI device carefully to avoid resource conflicts with devices that are not on the SCSI subsystem.

In summary, to understand SCSI hard drive technology think of it as a SCSI bus system—a closed bus system that can include several devices as well as the host adapter that acts as the bridge to the system bus. SCSI drives are usually faster than IDE drives but also more expensive. When buying SCSI drives, determine if the host adapter is built into the drive (embedded SCSI), or if you must purchase it separately. When installing a new SCSI system, be aware of the many variations of SCSI and buy compatible components.

Although the installation of a SCSI system may sound complicated and requires many decisions about what components to buy, the installation instructions for SCSI devices and host adapters are usually very thorough and well written. If you carefully follow all instructions, SCSI installations can be smooth and problem-free.

Comparing SCSI Hard Drives and EIDE Hard Drives

Consider the following issues when choosing between using an EIDE hard drive and a SCSI hard drive:

- A SCSI hard drive with its supporting host adapter and cable costs more than an EIDE hard drive with its supporting adapter card.
- A SCSI subsystem provides faster data transfer than an EIDE drive, although the SCSI bus is the source of the performance rather than the hard drive technology.
- A SCSI bus supports multitasking, allowing the CPU to request data from more than one SCSI device at the same time, whereas when the CPU requests data from an EIDE drive on an ISA bus, it can only process data from one I/O device at a time. The CPU must wait until the ISA bus and EIDE drive have completed the request before it can tackle another task. With SCSI, the CPU can perform another I/O task while waiting for the SCSI bus to complete the first request.
- A good SCSI host adapter allows you to connect other SCSI devices to it, such as a printer, scanner, or tape drive.
- Without SCSI technology, if you have two IDE drives on the same adapter, only one of them can be busy at any one time. For instance, without SCSI, if one of your IDE devices is a CD-ROM, the hard drive must wait for the CD-ROM to complete a task before it can work again. With SCSI, two or more devices can operate simultaneously. If you plan to transfer a lot of data from CD-ROM to hard drive, this is a good reason to choose SCSI.

In summary, SCSI is more expensive than EIDE but gives you better performance. Chapter 7 covers installation of SCSI devices in detail.

Other Types of Hard Drive Interfaces

A+ CORE 1.1, 1.7

In addition to the IDE and SCSI interface to the system bus, a hard drive can also interface with the system bus using IEEE 1394 and Fibre Channel. IEEE 1394, also known as FireWire and i.Link (named by Sony Corporation), was discussed in Chapter 3. It uses serial transmission of data and is popular with multimedia and home entertainment applications. For example, Quantum Corporation, a large hard drive manufacturer, makes a hard drive designed for home entertainment electronics that uses 1394 for the hard drive interface (Quantum in cooperation with Sony Corporation, calls it i.Link in the hard drive documentation). Another example of 1394 providing the interface for a hard drive is fireLINE External HotDrive by Evergreen Technologies. This external hard drive is intended for general-purpose storage and connects to a PC through a 1394 port provided either directly on the system board or by way of a 1394 expansion card. System board manufacturers have been slow to provide 1394 support on their system boards, mostly favoring support for USB instead. Generally, IDE is the slowest, SCSI is mid-range, and 1394 is the fastest, with some overlaps in these speeds. For a system to use 1394, the operating system must support it. Windows 98 and Windows 2000 support 1394, but Windows 95 and Windows NT do not.

6

Fibre Channel is another type of interface that can support hard drives. Fibre Channel is designed for use in high-end systems that have multiple hard drives. It competes with SCSI for these high-end solutions. As many as 126 devices can be connected to a single Fibre Channel bus as compared to 16 SCSI devices, including the host adapter. Fibre Channel is faster than SCSI when more than five hard drives are strung together to provide massive secondary storage, but is too expensive and has too much overhead to be a good solution for the desktop PC or workstation.

How a Hard Drive Is Logically Organized to Hold Data

Recall that today's hard drives come from the factory already low-level formatted (that is, having track and sector markings already in place). During installation, after the hard drive is physically installed, the next step is to partition the drive into manageable areas. The high-level divisions are called **partitions**, and within the partitions, the drive is further divided into logical drives or volumes. This section discusses the different types of division and how they are organized and used by the OS.

A+ CORE 1.2

Preparing a hard drive to hold data requires the following three steps:

1. **Low-level format**. This physically formats the hard drive and creates the tracks and sectors. For hard drives today, this has already been done by the time you buy the drive, and does not involve an OS.

A+CORE 1.2

2. **Partitioning the hard drive**. Even if only one partition is used, this step is still required. The FDISK program of Windows 9x or DOS sets up a partition table at the beginning of the hard drive. This table lists how many partitions are on the drive and their locations, and which partition is the boot partition. Within each partition, FDISK also creates logical drives, assigning letters to these drives.
3. **High-level format**. This can be done by DOS, Windows 9x, or some other OS for each logical drive on the hard drive. As each logical drive is formatted, the OS creates an OS boot record, a root directory, and the copies of FAT for the logical drive. (With floppy disks, the high-level format also creates the tracks and sectors, but with hard drives this has already been done by the low-level format.)

Hard Drive Partitions

Although you might have a 4-GB hard drive that is only a single physical drive, an OS can divide this single physical drive into more than one logical drive, which is called **partitioning the drive**. Two kinds of divisions take place. First the physical drive is divided into one or more partitions, and then each partition is further divided into logical drives or volumes. (A logical drive is sometimes called a logical partition; don't let the two uses of the term "partition" confuse you; partitions and logical partitions are divisions at different levels.) Figure 6-10 shows a typical example; the hard drive is divided into two partitions. The first partition contains one logical drive (drive C) and the second partition is divided into two logical drives (D and E). The **partition table** at the very beginning of the drive records all these divisions contained in the master boot sector (located in the very first sector at the beginning of the hard drive, on head 0, track 0, sector 1). Table 6-3 lists the contents of a partition table. Don't confuse the first physical sector of the hard drive with sector 1 as DOS or Windows 9x knows it. The OS's sector 1 comes after the physical sector 1.

The partition table is exactly 512 bytes long. During POST, the partition table program, sometimes called the **master boot record**, executes, checking the integrity of the partition table itself. If it finds any corruption, it refuses to continue execution, and the disk is unusable. If the table entries are valid, this program looks in the table to determine which partition is the active partition, and it executes the boot program in the boot record of that partition.

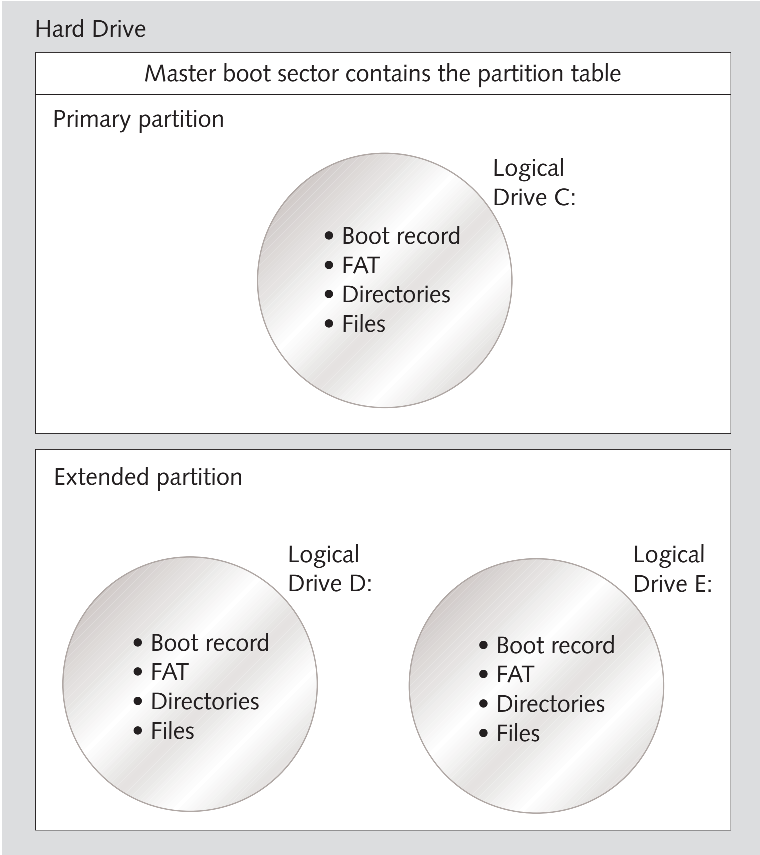

Figure 6-10 A hard drive is divided into one or more partitions that contain logical drives

Table 6-3 Hard drive partition table

Item	Bytes Used	Description
1	446 bytes	Program that calls the boot program on the OS boot record
2	16-byte total	Description of first partition
	1 byte	Is this the bootable partition? (Yes = 90h, No = 00h)
	3 bytes	Beginning location of the partition
	1 byte	System indicator; possible values are: 0 = Not a DOS partition 1 = DOS with a 12-bit FAT 4 = DOS with a 16-bit FAT 5 = Not the first partition 6 = Partition larger than 32 MB
	3 bytes	Ending location of partition
	4 bytes	First sector of the partition table relative to the beginning of the disk
	4 bytes	Number of sectors in the partition

Table 6-3 Hard drive partition table (continued)

Item	Bytes Used	Description
3	16 bytes	Describes second partition, using same format as first partition
4	16 bytes	Describes third partition, using same format as first partition
5	16 bytes	Describes fourth partition, using same format as first partition
6	2 bytes	Signature of the partition table, always AA55

Using DOS or Windows 9x, a hard drive can have only one primary partition and one extended partition, although the partition table can contain four partitions. Also, the primary partition can only have a single logical drive. In that case, the one logical drive in the primary partition is the only logical drive on the hard drive that can boot the operating system. The extended partition can have several logical drives.

The partition table must be created when the drive is first installed. This is done with the DOS or Windows 9x FDISK command or third-party software such as Partition Magic. Chapter 7 describes partitioning a hard drive in more detail.

A+ OS 1.3

When the drive is partitioned, FDISK assigns a drive letter to each logical drive. If only one logical drive is assigned, and this is the first hard drive installed in the system, this drive is called drive C. FDISK first creates a partition and then creates logical drives within the partition. You designate how many logical drives you want and how large they will be.

Logical Drives

Recall that both DOS and Windows 9x store data in files, each of which is allocated a group of whole clusters and cannot share a cluster with another file. The OS knows only the cluster numbers that it has assigned to a file; it does not track where these clusters are physically located. The OS keeps information about files and the clusters assigned to them in its file system. This arrangement is similar to a library where the readers are not allowed in the book stacks but must depend on a librarian to fetch the books for them (see Figure 6-11). When the OS requests BIOS to retrieve a file from the hard drive, BIOS, either on the system board or on the hard drive controller, determines where on the drive these sectors are located.

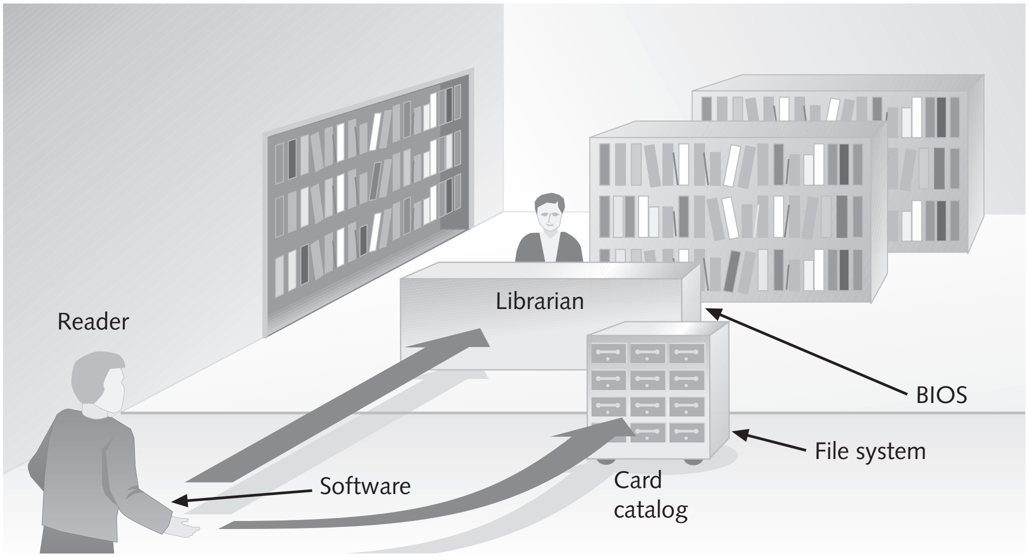

Figure 6-11 In some libraries, the reader (software) is not allowed in the stacks and depends on the librarian (BIOS) to know where to locate a book

A **logical drive** is a portion of a hard drive that an OS views and manages as an individual drive, much as it manages a floppy disk. The hard drive in Figure 6-10 is divided into three logical drives: drives C, D, and E. For a 3-GB drive, drive C might contain 2 GB, drive D 300 MB, and drive E 700 MB, to account for the entire physical drive capacity.

For the configuration in Figure 6-10, the three commands used to format these three logical drives are:

```
Format C:/S
Format D:
Format E:
```

The OS format for each logical drive creates these file system items at the beginning of each logical drive:

- Boot record
- FAT
- Root directory

These items are discussed below.

The Boot Record

A boot record is used during the boot process to inform the OS how a logical drive is organized. If the logical drive is the boot device, the boot program at the end of the boot record will load the hidden files (IO.SYS and MSDOS.SYS for DOS, and Msdos.sys for Windows 9x) during booting. Table 6-4 shows the complete record layout for the boot record. The

medium descriptor byte tells the OS what type of disk this is. The values of this descriptor byte are given in Table 6-5.

TIP The program in the boot record used to load the operating system is called the bootstrap loader; hence the phrase "booting the PC."

Table 6-4 Layout of the boot record

Description	Number of Bytes
Machine code	11
Bytes per sector	2
Sectors per cluster	1
Reserved	2
Number of FATs	1
Number of root directory entries	2
Number of logical sectors	2
Medium descriptor byte	1
Sectors per FAT	2
Sectors per track	2
Heads	2
Number of hidden sectors	2
Total sectors in logical volume	4
Physical drive number	1
Reserved	1
Extended boot signature record	1
32-bit binary volume ID	4
Volume label	11
Type of file system (FAT12, FAT16, or FAT32)	8
Program to load operating system (boot strap loader)	Remainder of the sector

Table 6-5 Disk type and descriptor byte

Disk Type	Descriptor Byte
3½-inch double-density floppy disk, 720K	F9
3½-inch high-density floppy disk, 1.44 MB	F0
Hard disk	F8

The FAT and the Root Directory

The OS uses two tables (the FAT and a directory, as seen in Figure 6-12) to keep track of which clusters are used for a file and other file information, including the filename and length, and whether the file is read-only or hidden. This high-level view of the file is all the OS needs to know. The physical location of the file is tracked by the BIOS or device driver managing the hard drive. The FAT and a directory are the vehicles of exchange between the

OS and the hard drive BIOS. The OS uses only one FAT for an entire logical drive but can have more than one directory on the drive. The main or root directory always present on a drive can have directories within it, called subdirectories in DOS or folders in Windows 9x.

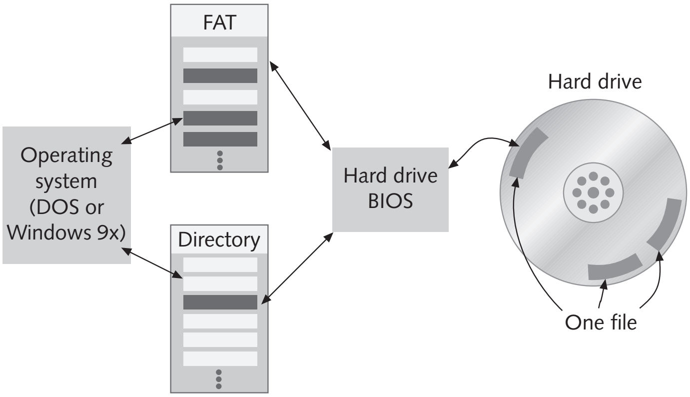

Figure 6-12 How the operating system views the hard drive when managing a file

To the OS, each logical drive looks like and is treated like a single floppy disk in this respect: as far as DOS or Windows 9x is concerned, a physical drive divided into three logical drives, C, D, and E, is equivalent to three separate physical drives (see Figure 6-13). The reason for this is that the OS manages a logical drive from the same high-level view whether it is a floppy drive, a part of a physical drive, or an entire physical drive. Each includes a FAT, one or more directories, and files that the OS tracks by using these tables.

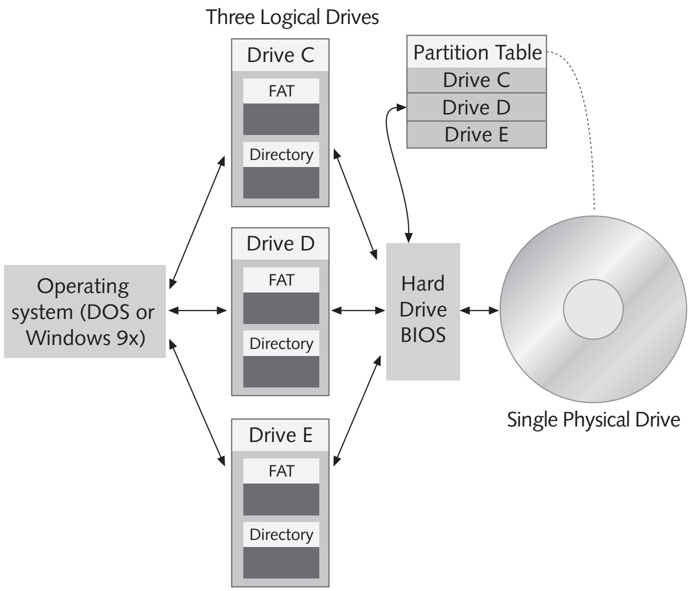

Figure 6-13 A single physical drive can be viewed by the operating system as one or more logical drives

A+ OS 1.3

For DOS and sometimes Windows 9x, each entry in a FAT for most hard drives is 16 bits, or a convenient 2 bytes, and the FAT is called FAT16. Besides FAT16, Windows 98 offers a FAT with 32-bit entries, and Windows NT offers an entirely different method of managing clusters called NTFS (new technology file system), which is discussed in Chapter 13. Windows 2000 supports FAT16, FAT32, and NTFS file systems. Windows 95 offers a version of FAT16 that can use long filenames, called virtual FAT or VFAT.

Remember that, as with floppy disks, each entry in a hard drive FAT tracks the use of one cluster. The number of sectors per cluster varies from one logical drive to another. Use the CHKDSK command to display the size of one cluster. There is another way to determine the size of a cluster with a simple test. First, use the DIR command and note how much space is available on your hard drive. Then, create a text file containing only a single character. Using the DIR command again, note how much disk space is available and compare the two values, before and after a single one-character file is written to the disk. The difference in the two values is the size of one cluster, which is the smallest amount that can be allocated to a file. In the FAT entries discussed below, we are using a hard drive that has one cluster equal to four sectors. A cluster is, therefore, 512 bytes/sector $\times$ 4 sectors/cluster, or 2,048 bytes.

Virtual File Allocation (VFAT) Windows 95 and Windows for Workgroups feature some improved methods of hard drive access, called **VFAT**, or **virtual file allocation**. These enable Windows to use 32-bit protected-mode device drivers for hard drive access. In Windows for Workgroups, VFAT is called 32-bit file access. Windows 95 supports filenames up to 255 characters. As you learned in Chapter 5, the filename and extension are stored in the root directory or in a subdirectory list. Each entry in the directory is 32 bytes long, and each 32-byte entry is called a block. Long filenames require more than one block in the directory. The FAT is not affected, but still uses 16 bits per cluster entry.

Some DOS-based disk utility programs can damage the entries in a directory in these additional blocks because they are not programmed to manage the extra blocks used to hold long filenames. Even a simple DEL command under OS/2 can leave the extra blocks in the directory used to hold the long filename unavailable for later use. The Windows 9x ScanDisk utility can recover these unreleased blocks.

A+ OS 1.3

FAT32 Beginning with Windows 95, Service Release 2 (sometimes called Windows 95b or Windows 95 OSR2), Microsoft offered a FAT that contains 32 bits per FAT entry instead of the older 12-bit or 16-bit FAT entries. Actually, only 28 of the bits are used to hold a cluster number; the remaining 4 bits are reserved.

The 32-bit FAT allows better management of very large hard drives, because the number of clusters per logical volume can increase: The largest cluster number a 16-bit FAT entry can hold is 65,535. The value of a 16-bit number when all 16 bits are 1s, that is, 1111 1111 1111 1111, is 65,535, which is then the largest number of clusters that the OS can support on a single logical drive. The largest logical drive capacity is dependent on the size of a single cluster and the number of clusters that can be accessed through the FAT. For FAT16, logical drives can range in size from 16 MB to 2,048 MB. But in order to get the larger sizes, the cluster size must be very large, sometimes as large as 32K, which can result in a lot of wasted space

A+ OS 1.3

for a hard drive that holds many small files. This wasted space is called **slack**. FAT32 makes it possible for cluster sizes to be smaller and still have very large logical drives, because there can be many more clusters to the drive. FAT32 is recommended for hard drives larger than 512 MB and is efficient for drives up to 16 GB. In this range, the cluster size is 8K. After that, the cluster size increases to about 16K for drives in the 16 GB to 32 GB range. You are then reaching a hard drive size that warrants a more powerful file management system than FAT32, such as NTFS supported by Windows NT and Windows 2000, discussed in Chapter 13.

If you are currently using FAT16 and are considering switching to FAT32, you can use Partition Magic, discussed in Chapter 7, to scan your hard drive and tell you how much of the drive is used for slack space. Knowing this can help you decide if the change will yield you more usable drive space. In Chapter 12 you will learn how to set up a new hard drive to use FAT32 or convert an older hard drive to it.

The Root Directory The layout of the root directory is the same for hard drives as for floppy disks, discussed in Chapter 5. Recall that the total number of bytes for each file entry in a directory is 32. Refer back to Chapter 5 for a description of each entry in the root directory.

The maximum number of entries in the root directory for DOS and Windows 95 is fixed. The number of entries in a subdirectory, however, is not limited. The fixed reserved length of the root directory for current versions of DOS and Windows is 512 entries; early versions of DOS didn't allow as many entries in the root directory. Note, however, that the OS manuals recommend that you keep only about 150 entries in any one directory. Having any more entries slows access to the directory. The number of entries in the root directory is stored in the boot record of the hard drive. Using long filenames reduces the number of files that can be stored in this fixed number of entries in the root directory, because the long filenames require more than one entry in the directory. Windows 98 does not limit the size of the root directory.

Communicating with the Hard Drive BIOS

Recall that, beginning with IDE hard drives, the number of sectors per track varied from one track to another. Therefore, the OS and system BIOS could not count on using actual hard drive cylinder, head, and sector coordinates when making requests for data through the hard drive BIOS (which is how requests were made with early hard drives). Instead, sophisticated methods were developed so that system BIOS and the OS could communicate with the hard drive controller BIOS in familiar terms, but only the controller BIOS knew where the data was physically located on the drive.

In order to build a foundation for understanding these more complicated solutions, we begin our discussion of communicating with hard drive BIOS by looking at BIOS that uses older hard drive technology (using a consistent relationship among cylinder, head, and sectors on a hard drive). Looking at Figure 6-14, you can see the different stages in communication with these older hard drives. The OS and other software communicated the cylinder, head, and sector coordinates to system BIOS, which communicated them to the hard drive controller BIOS, which used these coordinates to locate data on the hard drive. All levels communicated the same cylinder, head, and sector information to locate data on the hard drive. No tricks, just

straightforward communication. With this straightforward communication, drive capacity could easily be calculated and drive access was simple, when drives were small and all tracks on the drive had the same number of sectors. Subsequently, methods used to communicate among the different levels were altered to accommodate larger drives and new drive technology.

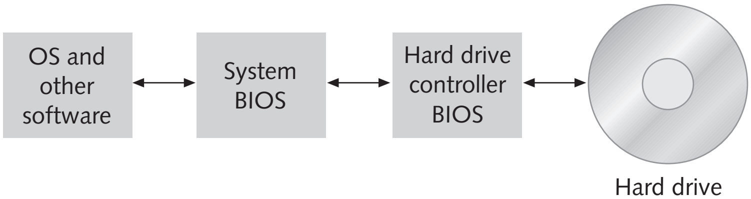

Figure 6-14 With older hard drives, cylinder, track, and sector information was communicated at each level

Calculating Drive Capacity

How much data can be stored on a hard drive? Back when hard drives used a constant number of sectors per track, measuring drive capacity was straightforward. Recall that the OS views the data as groups of sectors of 512 bytes each. The number of sectors present on the drive determined the drive capacity. Each surface or platter on a hard drive is divided into tracks and sectors. All sectors in a track hold 512 bytes regardless of the radius of the track. If you knew the number of tracks, heads and sectors per track, you could calculate the storage capacity of a drive, because all tracks had the same number of sectors. Software and operating systems were written to interface with BIOS, which managed a hard drive by this method of assuming that for each hard drive there was a fixed relationship among the number of sectors, tracks, heads, and cylinders.

The table in Appendix B, "Hard Drive Types," lists the drive types, first established by IBM and later added to by other companies, that were supported by system BIOS. Notice in Appendix B that most earlier and smaller drives have 17 sectors per track. Later in the list are a few drives that have 26 sectors per track (see types 33, 34, and 37).

Because some drives are not on the list, most system BIOS programs permit a user-defined hard drive type. When you choose a user-defined drive type, you must tell the setup the number of heads, tracks, and sectors that the drive has, as well as some other information, so that system BIOS knows how to address the drive. These drive types might seem like ancient history with respect to hard drive capacity today, but the concepts and calculations apply to today's modern drives in a similar fashion.

TIP

When installing a hard drive, it was once necessary to tell CMOS setup the drive capacity. Today, most system BIOS offers **autodetection**, a method whereby the BIOS will detect the new drive and automatically select the correct drive capacity and configuration for you.

To learn how drive capacity is calculated with these older drive layouts, let's calculate the storage capacity of drive type 12 in Appendix B. The following table contains the information needed to calculate capacity:

Type	Number of Tracks	Number of Heads	Number of Sectors/Track
12	855	7	17

When you see an odd number of heads, you know that the first head is not used for data. The drive listed with seven heads really has eight surfaces, or four platters, with the top surface of the top platter used to record the layout information for the entire drive. You calculate the capacity of the drive as follows:

855 tracks × 17 sectors/track × 512 bytes/sector = 7,441,920 bytes

There are 7,441,920 bytes on one surface, or head, of the drive. Therefore, the drive capacity is as follows:

7,441,920 bytes/head × 7 heads = 52,093,440 bytes

To convert bytes to kilobytes, divide by 1,024, as follows:

52,093,440 bytes × 1 KB/1,024 bytes = 50,872.5 KB

To convert kilobytes to megabytes, divide by 1,024 as follows:[2]

50,872.5 KB × 1MB/1,024 KB = 49.68 MB capacity

Let's work through one more example, for drive type 37. From Appendix B, the specifications are as follows:

Type	Number of Tracks	Number of Heads	Number of Sectors/Track
37	1,024	5	26

The capacity of this drive is calculated as follows:

1,024 tracks × 26 sectors/track × 512 bytes/sector × 5 heads = 68,157,440 bytes

68,157,440 bytes/1,024/1,024 = 65 MB

Adjusting for More Complex Hard Drive Organization

As hard drive size and technology improved, the OS and other software required new methods to relate to hard drive BIOS because of two situations that arose:

- Beginning with IDE technology, the number of sectors per track varied depending on the location of the track, which made it impossible for the OS and software to address the data on the hard drive using actual cylinder, head, and sector parameters.

[2] For hard drive capacity, some tables use the relation 1 MB = 1,000K and others use 1 MB = 1,024K. There are no standards; in fact, sometimes both relationships may be used in the same table to calculate drive capacity. A close look at Appendix B demonstrates the lack of consistency.

6

- When hard drives were small, the maximum size of the parameters that the OS and software sent to hard drive BIOS was established. These maximum values placed self-imposed limitations on the size hard drive that software can address using actual cylinder, head and sector parameters.

A+ CORE 2.1, 4.4

However, it was important for the industry to retain backward compatibility so that legacy operating systems and other software could work. As is common in the evolution of computers, clever methods were devised to "trick" older technology so that it could work in newer environments. The older, legacy technology (in this case, software) still sees its world unchanged because the newer technology (in this case, BIOS) shelters it from the new methodology. These "deceptions" can happen at several stages of communication, in the following ways:

- The hard drive can use a complex cylinder, head, and sector organization that only the controller BIOS knows. However, the controller BIOS communicates to system BIOS in terms of the older methodology. When this method is used, the actual organization of the hard drive is called the physical geometry of the drive, and the organization communicated to system BIOS is called the logical geometry of the drive. This method is called the **CHS mode** (cylinder, head, sector), or **normal mode**.
- The hard drive controller BIOS sends the logical geometry to system BIOS, but system BIOS communicates a different set of parameters to the OS and other software. This method is called **translation**, and system BIOS is said to be in **large mode**, or **ECHS mode** (**extended CHS**).
- The hard drive controller BIOS and system BIOS communicate using a method entirely different from cylinder, head, and sector information. System BIOS sends cylinder, head, and sector information to the software, which is neither logical nor physical geometry. This method of translation is referred to as **LBA mode** (**logical block addressing**).
- The OS and software can bypass the system BIOS altogether and communicate directly with the controller BIOS by using device drivers. This is the method used by Windows NT and Windows 2000. True to its compromising nature, Windows 9x has its own 32-bit protected mode device drivers to access hard drives, bypassing system BIOS. However, in order to support DOS and other older software, Windows 9x also supports using system BIOS to access drives.

Physical Geometry and Logical Geometry

Although today's hard drives no longer organize data on the drive in a straightforward manner (Figure 6-4), but rather use zone bit recording (Figure 6-5), drive capacity is calculated in the same way, as if the number of sectors per track were constant. Controller BIOS masks the actual organization of the drive, called its **physical geometry**, from system BIOS and software. It communicates to system BIOS and software a number of cylinders, heads, and sectors, which, when used in calculations, yields the actual capacity of the drive, although the physical geometry may be quite different. These bogus values for cylinders, heads, and sectors are called the **logical geometry** of the drive.

CHS or Normal Mode

Recall that CHS mode (cylinders, heads, sectors), or normal mode, is the method system BIOS uses to manage a hard drive—it communicates to the controller BIOS the logical geometry of the hard drive based on tracks (or cylinders), heads, and sectors. This was the only method available for several years. CHS mode can be used only on hard drives where the following is true:

- The number of cylinders does not exceed 1,024.
- The number of heads does not exceed 16.
- There are no more than 63 sectors/track, assuming a constant number of sectors/track.
- There must be 512 bytes per sector.

Based on these criteria, the maximum amount of storage on a hard drive using CHS mode is 528 MB or 504 MB, depending on how the calculations are done (1K = 1000 or 1K = 1024). This limitation exists because of a combination of two factors: the way software calls up system BIOS to access the drive and the IDE/ATA interface standard that many hard drives adhere to (see Figure 6-15).

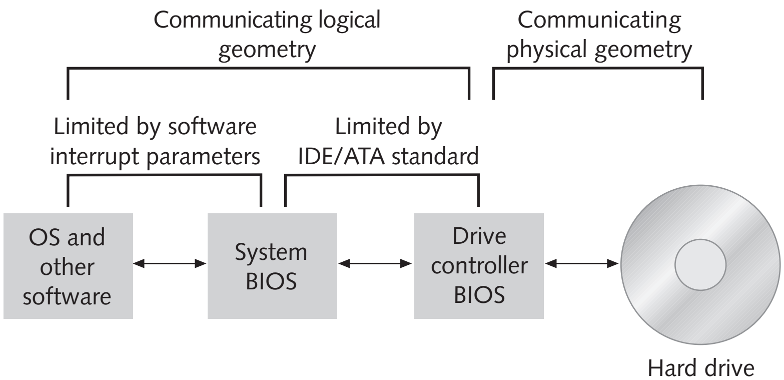

Figure 6-15 In using CHS mode to access a hard drive, limitations exist at two stages of communication, resulting in a combined limitation of 504 MB

Recall from Chapter 2 and Appendix G that software calls BIOS by issuing a software interrupt. The interrupt number for input and output to hard drives is Int 13h. At the same time, software also sends to the interrupt request handler (BIOS) the CHS parameters giving the location of the data it wants to access. A decision was made early in the evolution of managing hard drives to allot these parameters a certain number of memory addresses, which limits the maximum values for these parameters (see Table 6-6). Notice in the last column of Table 6-6 that if all bits are ones (giving the largest possible values), then 7.88 GB is the largest possible drive supported by this process whereby software uses interrupt 13h to call system BIOS.

Table 6-6 CHS parameters used by a software interrupt for system BIOS to access a hard drive

Description	Cylinders	Heads	Sectors	Totals
Number of bits used to store the value	10 bits	8 bits	6 bits	24 bits
Value range	0 to 1023	0 to 255	1 to 63	NA
Maximum number of values	1024 cylinders	256 heads	63 sectors	7.88 GB

The second limitation is a result of the IDE/ATA interface standards for hard drives for communication between the hard drive controller BIOS and system BIOS. When IDE hard drives were first built, different manufacturers used different interface standards to communicate the logical geometry of their drives to system BIOS. This resulted in the possibility that two drives from two different manufacturers could not coexist using only one system BIOS. To eliminate this compatibility problem, the IDE/ATA standard was adopted by the **American National Standards Institute** (**ANSI**), a nonprofit organization dedicated to creating trade and communications standards. The IDE/ATA interface standard maximums are listed in Table 6-7. If all 28 bits are ones, the drive size is 128 GB, the largest size that the IDE/ATA standard supports.

Table 6-7 Limitation of the IDE/ATA standard for hard drives

Description	Cylinders	Heads	Sectors	Total
Number of bits used to store the value	16 bits	4 bits	8 bits	28 bits
Maximum number of values	65,536 cylinders	16 heads	256 sectors	128 GB

The software interrupt Int 13h supports 7.88 GB, and the IDE/ATA standard supports 128 GB. Unfortunately, each standard distributed bits differently, so communication across both standards is limited to the smaller value of the two parameters. Therefore, the maximum values allowed using both standards are 1,024 cylinders (smaller value comes from Int 13h), 16 heads (smaller value comes from IDE/ATA standard), and 63 sectors (smaller value comes from Int 13h), which give us the 504 MB limitation. It's amazing that this limitation is not at all a limitation of hard drive technology, but rather a limitation caused by poor collaboration of standards. Figure 6-16 shows the resulting overall limitation that is caused by a combination of the limitations of the two standards.

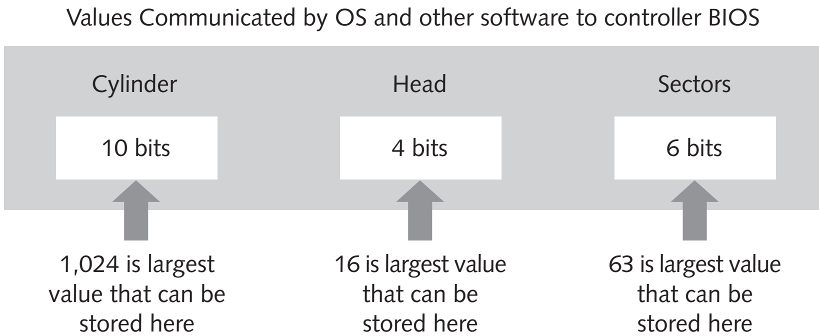

Figure 6-16 In using CHS mode, the three limitations on the number of bits to transfer cylinder, head, and sector information are caused by software interrupt parameters and IDE interface limitations

Translation Methods

With hard drives larger than 504 MB (called large-capacity drives), translation is used to bypass these limitations in our standards and maintain backward compatibility with older software. System BIOS that supports translation methods and, therefore, allows access to drives larger than 504 MB is called **enhanced BIOS**. Remember that system BIOS supports a hard drive in one of three ways:

- CHS, or normal, mode (for drives less than 504 MB)
- ECHS mode, or large mode (uses translation to access drives between 504 MB and 8.4 GB)
- LBA mode (uses translation to access drives larger than 504 MB)

Look at the CMOS settings of your computer to determine which modes your system BIOS supports. In Figure 6-17, you can see where the two major translation methods occur in the different stages of communication with the hard drive.

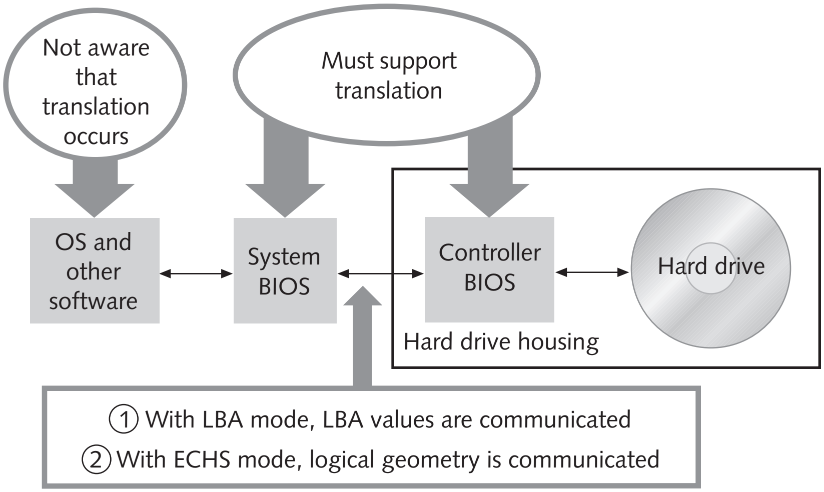

Figure 6-17 Translation methods must be supported by system BIOS and the hard drive controller BIOS within the hard drive housing

A+CORE 2.1

ECHS Mode or Large Mode For large-capacity drives, using ECHS mode, or large mode, system BIOS translates the logical geometry of the drive given to it by the controller BIOS into parameters that the software interrupt Int 13h can handle. As you can see from Table 6-6, Int 13h allocates the number of bits that allows for a very large value for the number of heads. Using ECHS, system BIOS adjusts the parameters to use a larger value for heads and a smaller value for cylinders than the original logical geometry. It does this to (1) make the calculation of drive capacity correct, (2) still adhere to the Int 13h limitation, and (3) break the 504-MB barrier. ECHS is sometimes called Extended Int 13h. Using this method, the maximum logical geometry of a drive is 16,383 cylinders, 16 heads, and 63 sectors yielding a drive capacity of 16,383 cylinders × 16 heads × 63 sectors × 512 bytes per sector = 8,455,200,700 bytes or an 8.4 GB drive. This 8.4 GB drive capacity is the maximum size hard drive that can be supported using a DOS or Windows 9x FAT16 file system and system BIOS using the ECHS mode. The 8.4 GB barrier is broken using LBA mode for system BIOS and FAT32 for the OS file system.

ECHS mode is not as popular as LBA mode. If you are not sure which mode your large-capacity hard drive supports and you need to make a selection in setup, LBA mode is your best guess.

LBA Mode In using LBA mode to support large-capacity drives, the hard drive controller BIOS does not present the logical geometry to system BIOS (based on cylinders, heads, and sectors), but rather communicates by using another method called logical block addressing (LBA). System BIOS simply views the hard drive as a long list of sequential numbers (0, 1, 2, 3,…) called LBAs or addressable sectors. If system BIOS is being used to interface with the software, it then uses a method of enhanced CHS to give to the software cylinder, head, and sector information consistent with the overall capacity of the drive. These parameters

A+ CORE 2.1

will have no relationship to the actual physical location of the data on the drive. LBA mode is the most popular way of dealing with drives larger than 504 MB and is the only way of dealing with drives larger than 8.4 GB.

For hard drives greater than 8.4 GB, CMOS setup reports a drive capacity in cylinders, heads, and sectors that does not reflect the true size of the drive. Because of this, the normal way of calculating the size of the drive does not work. For example, a 20 GB hard drive contains the following information in CMOS:

Cylinders: 1024
Heads: 255
Sectors: 63
CHS Capacity: 8422 MB
Maximum LBA capacity: 20525 MB

Though the cylinder, head, and sector information is incorrect, the LBA capacity reflects the true size of the drive.

TIP

To use the operating system to report the capacity of a hard drive, for DOS, enter the command CHKDSK at the DOS prompt. For Windows 9x, using Windows Explorer, right-click the drive letter and select Properties from the shortcut menu. Report the capacity of each logical drive on the hard drive and then add them together to get the entire hard drive capacity.

System BIOS Helps Manage Data Transfer

System BIOS is used in three different ways to support a hard drive:

- System BIOS provides the interrupt handler for software interrupts (discussed above).
- System BIOS can automatically detect and configure a hard drive (will be discussed in Chapter 7).
- System BIOS helps manage data transfer over the I/O bus between the hard drive and memory.

So far in our discussion, we have considered how data is accessed on the drive by using various methods of viewing the data on the drive. Now we turn our attention to how the data is transferred over the bus to memory, once the data is accessed. There are several data transfer methods and protocols used, many of which must be supported by system BIOS as well as the hard drive, in order to work. As hard drive technologies improve, so do the standards managing them. There is IDE/ATA, the first hard drive standard, followed by ATA-2, Fast ATA, Fast ATA-2, ATA-3, ATAPI, Ultra ATA, Ultra ATA/66, and Ultra ATA/100. In most cases, a PC technician does not need to know which standard a hard drive supports, because BIOS autodetection selects the best possible standard supported by both hard drive and BIOS.

Who's in Charge? Data transfer over the I/O bus can be managed in one of three ways, depending on which component controls the process.

- Using programmed I/O (PIO) mode, the CPU is in charge and manages data transfer. There are five different PIO modes (0 to 4) with transfer rates from 3.3 MB/sec to 16.6 MB/sec.
- Using DMA with the DMA controller in charge, data is transferred to memory without the involvement of the CPU.
- With bus mastering using DMA, the hard drive BIOS controls data transfer. The highest transfer rates are attained using this method. Transfer rates using DMA with bus mastering are 33, 44, 66, and 100 MB/sec. Speeds of 66 MB/sec and higher require the special 80-conductor IDE cable. Error checking is also done.

The latest advancement in data transfer is Ultra ATA (also called Ultra DMA) which involves bus mastering controlled by the hard drive BIOS. The hard drive controller, the system board chipset, and the operating system must support the transfer mode. For more information about this technology, see the web site of T13 Committee, the organization responsible for its standards, at www.t13.org.

How Much Can be Transferred at One Time? Using the IDE/ATA standard, data is transferred 16 bits at a time from the hard drive to the I/O bus over the hard drive cable. Once on the bus, data can be transferred at 16-bit access or 32-bit access.

How Much Overhead? Some BIOSs support a method of data transfer that allows multiple transfers of data on a single software interrupt. This method is called **block mode** and should be used if your hard drive and BIOS support it. However, if you are having problems with hard drive errors, one thing you can try is disabling block mode. Use CMOS setup to make the change.

Operating System Commands to Manage a Hard Drive

DOS and Windows 9x have similar commands and methods of managing a hard drive. Below, DOS commands are discussed first, and then the same functions are covered in Windows 9x.

DOS Commands to Manage a Hard Drive

The DOS commands to manage a hard drive are described in this section, along with examples of how to use them. For more extensive discussions, see the *DOS User's Guide and Reference Manual*, or at the DOS prompt type HELP followed by the name of the command. In addition to the commands below, other DOS commands covered in Chapter 5 to manage files on a floppy drive can also be used on a hard drive. These are FORMAT, LABEL, DEL, UNDELETE, RECOVER, COPY, XCOPY, and DELTREE.

MKDIR [drive:]path or MD [drive:]path Command

The **MKDIR** (abbreviated MD, standing for make directory) **command** creates a subdirectory entry in a directory. This command can be used on both hard drives and floppy disks. Think of a subdirectory as a list within a list, much like the hierarchy of an outline or a company organizational chart. In an outline, there are major categories, and under some of these major categories more detailed subordinate items are listed. Parent directories are equivalent to the major categories. These parent directories can have subordinate directories, or subdirectories (sometimes called child directories) listed within them. When you move to these subdirectories, you see another list. This list too can have subdirectories.

You can use subdirectories to organize and categorize files. Long lists of unrelated items are tedious and difficult to manage. Putting related files under a single subdirectory heading simplifies your work. Software installation programs place an application it is installing in a separate directory to prevent mixing its program files with other files on the hard drive. Data that is created by a software package should not be put in the same directory as the software itself, for several reasons. If you are creating, copying, or deleting data files, you are less likely to erase or overwrite a software file if the data is in a different directory. Also, there is no question later about which files are data files and which are software files.

Another reason to keep data in one directory and the software in another is that it makes backing up easier. As soon as you have installed a software package and confirmed that it is working properly, you may want to back up your hard drive. This one backup of the software is all that's necessary unless you make a change to the software. The data, however, needs backing up as soon as you have several hours invested in it. If the data changes often, make regular backups. How often depends on how much data is changed and when it is changed. If the data is in its own directory, you can easily back up just that directory instead of having to back up the software with the data.

It's good practice to create a data directory in the software directory that manages that data. The software directory is called the parent directory, and the data directory is called a child directory. Just as with outlines, you commonly say that the child directory is "under," or subordinate to, the parent directory. When viewing the directory structure, it is easy, then, to see which data files belong to which software.

Another approach is to create one directory under the root directory named \Data. Put all data files under this directory, organized by type of data or by subject (see Figure 6-18). If you prefer to organize the data by file type, within the \Data directory, create one directory for each application program you are using and store the data for that application in that subdirectory. If you prefer to organize by subject, create a subdirectory for each subject (examples are personal, administration, Project 1, Project 2), and place all file types that pertain to that subject in that subdirectory. By putting all data files under a single subdirectory tree structure, it is easy to quickly back up all the data on your hard drive by selecting only this one directory, \Data. For example, to back up all the data on your hard drive to a Zip drive, select \Data and copy its contents to the Zip drive.

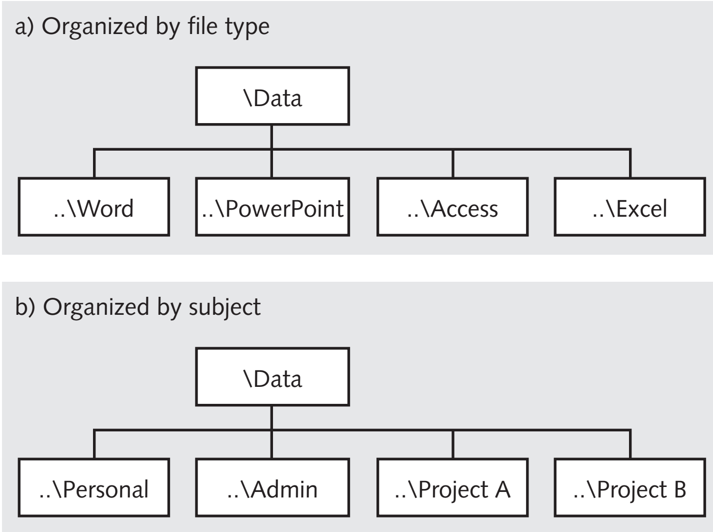

Figure 6-18 Two subdirectory trees to hold data on a hard drive

MKDIR Syntax and Some Examples When using the MKDIR command, you need not specify the drive if the drive is the current default drive, and you need not specify the parent directory if it is the current default directory. To create a directory named \GAME on drive C, for example, use this command:

```
MKDIR C:\GAME
```

The backslash indicates that the directory is under the root directory. To create a directory named CHESS under the \GAME directory, use this command:

```
MKDIR C:\GAME\CHESS
```

DOS requires that the parent directory GAME must already exist before it can create the child directory CHESS.

A+ OS 1.1

Figure 6-19 shows the result of the DIR command on the directory \GAME. Note the two initial entries in the directory table, the . (dot) and the .. (dot, dot) entries. These two entries are created by the MKDIR command when DOS initially sets up the directory. You cannot edit these entries with normal DOS commands, and they must remain in the directory for the directory's lifetime. The . entry points to the subdirectory itself, and the .. entry points to the parent directory—in this case, the root directory.

A+ OS 1.1

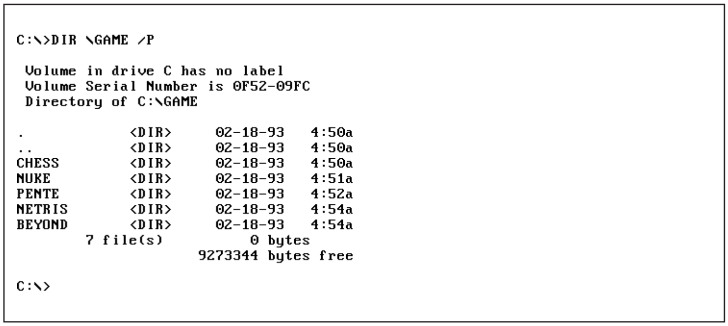

Figure 6-19 DIR of the \GAME directory

6

CHDIR [drive:]path or CD [drive:]path or CD..

The **CHDIR** (abbreviated CD, standing for change directory) **command** changes the current default directory. In its easiest-to-follow form, you simply state the drive and the entire path that you want to be current:

```
CD C:\GAME\CHESS
```

If you have previously issued the PROMPT PG command, either from the DOS prompt or from the AUTOEXEC.BAT file, the DOS prompt will look like this:

```
C:\GAME\CHESS>
```

To move from a child directory to its parent directory, use the .. variation of the command:

```
C:\GAME\CHESS> CD..
C:\GAME>
```

Remember that .. always means the parent directory. You can move from a parent directory to one of its child directories simply by stating the name of the child directory:

```
C:\GAME> CD CHESS
C:\GAME\CHESS>
```

Do not put a backslash in front of the child directory name; doing so tells DOS to go to a directory named CHESS that is directly under the root directory.

RMDIR [drive:]path or RD [drive:]path

The **RMDIR command** (abbreviated RD, standing for remove directory) removes a subdirectory. Before you can use the RMDIR command, three things must be true:

- The directory must contain no files.
- The directory must contain no subdirectories.
- The directory must not be the current directory.

The . and .. entries are present when a directory is ready for removal. For example, to remove the \GAME directory in the above example, the CHESS directory must first be removed:

```
C:\> RMDIR C:\GAME\CHESS
```

Or, if the \GAME directory is the current directory, use this command:

```
C:\GAME> RD CHESS
```

Once you remove the CHESS directory, you can remove the \GAME directory. You must first leave the \GAME directory like this:

```
C:\GAME>CD..
C:\> RD \GAME
```

TREE [drive:][path] [/F] [/A]

The **TREE command** displays the directory structure of a hard drive or disk. If you do not specify a drive or path, the TREE command displays the directory structure of the current drive and current directory. The command displays the structure in graphic form unless you include the /A option, which specifies text format. The /F option has the TREE command include the names of the files in a directory as well as the directory name.

For example, to display the subdirectory structure of the \Windows directory with files also listed, one screen at a time, use this command:

```
C:\> TREE \WINDOWS /F | MORE
```

The |MORE option at the end of the command line causes the output of the command to be displayed one screen at a time rather than continuously scrolled.

To send the output to a printer, use this command:

```
C:\> TREE \WINDOWS /F > PRN
```

A+ OS 1.1, 3.2

ATTRIB

The **ATTRIB command** displays or changes the read-only, archive, system, and hidden attributes assigned to files. To display the attributes of the file MYFILE.TXT, use this command:

```
ATTRIB MYFILE.TXT
```

To hide the file, use this command:

```
ATTRIB +H MYFILE.TXT
```

To remove the hide status of the file, use this command:

```
ATTRIB -H MYFILE.TXT
```

To make the file a read-only file, use this command:

```
ATTRIB +R MYFILE.TXT
```

To remove the read-only status of the file, use this command:

```
ATTRIB -R MYFILE.TXT
```

A+ OS 1.1, 3.2

To turn the archive bit on, use this command:

```
ATTRIB +A MYFILE.TXT
```

To turn the archive bit off, use this command:

```
ATTRIB -A MYFILE.TXT
```

MIRROR

The **MIRROR command** saves partition table information to disk when used with the /PARTN parameter, like this:

```
MIRROR /PARTN
```

6

The MIRROR command is an older DOS 5 command that was not included with later versions of DOS. If you have access to DOS 5, even after you upgrade to a later version of DOS, keep the command file for MIRROR in your DOS directory as a quick and easy method of saving partition table information to floppy disk.

UNFORMAT

The **UNFORMAT command** not only reverses the effect of an accidental format, but also repairs a damaged partition table if the table has previously been saved with the MIRROR /PARTN command.

To unformat a disk, use this command:

```
UNFORMAT C:
```

To repair a damaged partition table if the table has previously been saved to a disk, use this command:

```
UNFORMAT /PARTN
```

PATH

As discussed in Chapter 2, the PATH command lists where DOS and Windows 3.x should look to find executable program files. This command is discussed here again to make the list of commands more complete. A sample PATH command is

```
PATH C:\;C:\DOS;C:\WINDOWS;C:\UTILITY
```

Each path is separated from the next with a semicolon. You should put the most-used paths at the beginning of the line, because the OS searches the paths listed in the PATH command line from left to right. The PATH command goes in the AUTOEXEC.BAT file and can be executed from the DOS prompt.

Using Batch Files

Suppose you have a list of DOS commands that you want to execute several times. Perhaps you have some data files to distribute to several PCs in your office, and, having no LAN, you must walk from one PC to another doing the same job repeatedly. A solution is to store the list of commands in a **batch file** on disk and then execute the batch file at each PC. DOS

and Windows require that the batch file have a .BAT file extension. For example, store these five DOS commands on a disk in a file named MYLOAD.BAT:

```
C:
MD\UTILITY
MD\UTILITY\TOOLS
CD\UTILITY\TOOLS
COPY A:\TOOLS\*.*
```

From the C prompt, you execute the batch file, just as with other program files, by entering the name of the file with or without the file extension:

```
A:\>MYLOAD
```

All the commands listed in the file will execute, beginning at the top of the list. The batch file above will create a subdirectory under the C drive called utility\tools; change to that directory as the default directory, and copy all files from the disk in drive A into that new subdirectory. Look at any good book on DOS to find examples of the very useful ways you can elaborate on batch files, including adding user menus. AUTOEXEC.BAT is an example of a batch file.

Using Windows 9x to Manage a Hard Drive

Windows 9x Explorer is the primary tool for managing the files on your hard drive. Windows 9x calls a directory or subdirectory a **folder**. Windows 98 is used in the following examples, but Windows 95 works the same way. Open Explorer in Windows 98 (click **Start**, **Programs**, **Windows Explorer**, or right-click **My Computer** and select **Explore** from the menu) and follow the directions below to manage files and folders on your hard drive.

A+ OS 1.1

Create a New Folder

To create a folder (equivalent to the DOS MD command), do the following. Select the folder you want to be the parent folder by clicking the folder name. For example, to create a folder named Chess under the folder named Games, first click the Games folder. Then click the **File** menu. Select **New** from the menu. Then select **Folder** from the submenu that is displayed, as in Figure 6-20. The new folder will be created under Games, but its name will be New Folder. Click the folder name to open the text box and change the name of the folder. Change the name from New Folder to Chess, as in Figure 6-21.

A+ OS 1.1

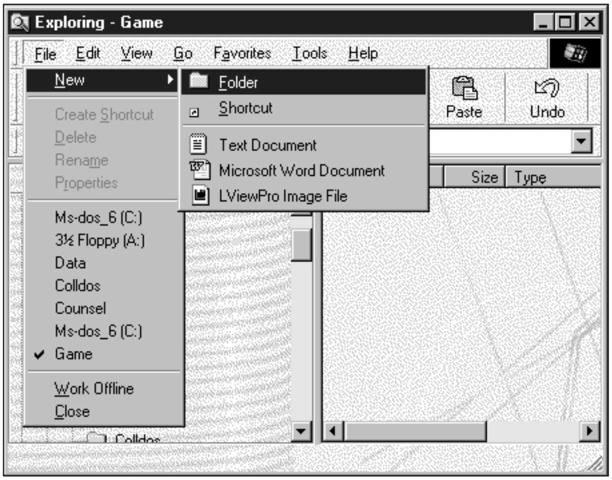

Figure 6-20 Create a new folder

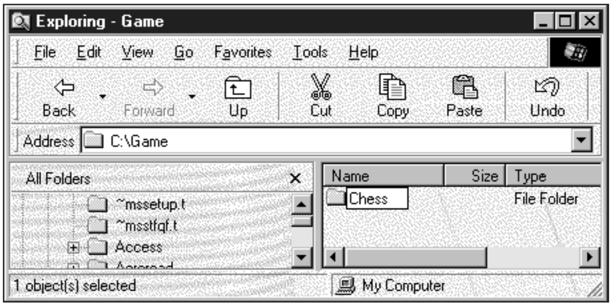

Figure 6-21 Edit the new folder's name

Delete a Folder

To delete a folder (equivalent to the DOS RD command) from Explorer, right-click the folder and select **Delete** from the shortcut menu. A confirmation dialog box like the one in Figure 6-22 asks you if you are sure you want to delete the folder. If you respond **Yes**, the folder and all of its contents, including subfolders, will be sent to the Recycle Bin. Empty the Recycle Bin to free your disk space.

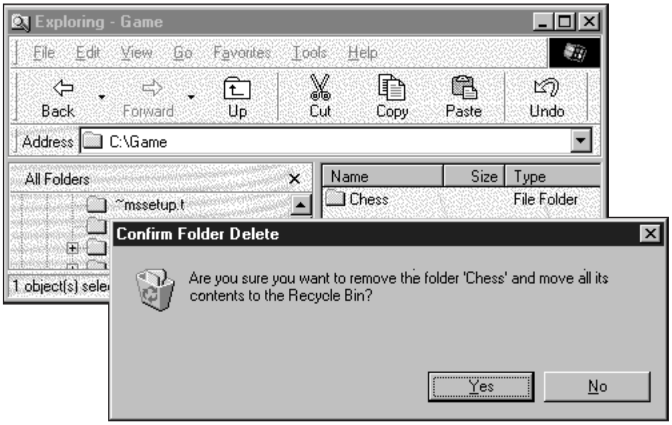

Figure 6-22 Delete a folder in Windows 98

A+ OS 1.2

File Properties

To access file properties in Windows 98, from Explorer, right-click a file and select **Properties** from the shortcut menu. The Properties window appears, as shown in Figure 6-23. Windows 98 identifies the file type primarily by the file extension. From the Properties window, you can change the attributes of the file.

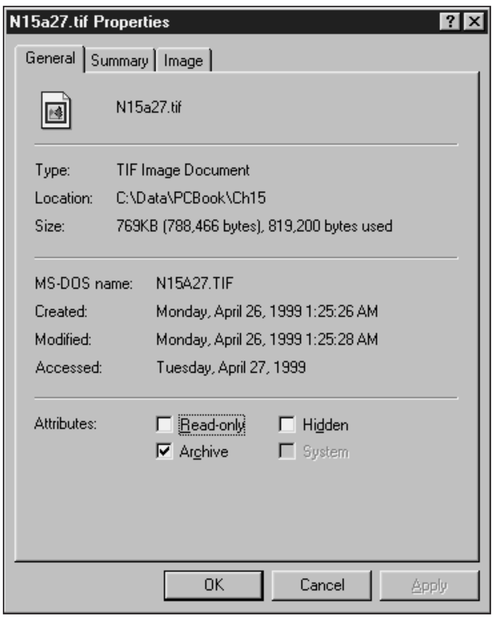

Figure 6-23 Properties of a file in Windows 98

The PATH Command and Batch Files

Although the PATH command and AUTOEXEC.BAT are not necessary in Windows 9x, you can use them the same way as in DOS and Windows 3.x. If you have an AUTOEXEC.BAT file in your root directory when Windows 9x starts, it reads the PATH command stored in that file. You can also store DOS commands in batch files and execute them from Windows 9x by double-clicking the filename of the batch file in Explorer.

Windows 9x uses a default path of C:\Windows; C:\Windows\Command if you don't have an AUTOEXEC.BAT file. Not all paths to program files must appear in the PATH command, as they do in DOS. To edit AUTOEXEC.BAT in Windows 9x, right-click the filename and select **Edit** from the shortcut menu. Notepad is executed. After making changes, restart your computer for the changes to take effect.

OPTIMIZING A HARD DRIVE

In the first part of this chapter, you saw how data is stored on a drive from the perspective of the OS, which sees the drive as a long list of clusters made up of 512-byte sectors. This section will delve more deeply into drive technology, looking at how a drive is formatted, the different ways data is stored, and some ways to optimize data access. Formatting a hard drive is introduced here in preparation for understanding some of the concepts of optimizing the drive discussed later in the chapter. A more detailed discussion of formatting a hard drive is left for Chapter 7, which covers installing a hard drive.

Fragmentation

A+ OS 1.3, 1.1, 3.2

Fragmentation is the undesirable placement of a single file in several cluster locations that are not right next to each other, so that data access time is increased. When a hard drive is new and freshly formatted, the OS writes files to the drive beginning with cluster 2, placing the data in consecutive clusters. Each new file begins with the next available cluster. Later, after a file has been deleted, the OS writes a new file to the drive, beginning with the first available cluster in the FAT. If the OS encounters used clusters as it is writing the file, it simply skips these clusters and uses the next available one. In this way, after many files have been deleted and added to the drive, files become fragmented. Fragmentation occurs when files are written to a drive in more than one group of contiguous clusters. The clusters that make up a file are together called a **chain**. For a well-used hard drive, it is possible to have a file stored in clusters at 20, 30, 40, or more locations. Fragmentation is undesirable because (1) when DOS has to access many different locations on the drive to read a file, access time slows down, and (2) if the file should become corrupted, recovering a fragmented file is more complicated than recovering a file in one continuous chain.

For these reasons, one routine maintenance task is to periodically **defragment** the hard drive. For DOS, the simplest way to do this is to use DOS 6+ DEFRAG or a utility software package, such as Nuts & Bolts or Norton Utilities, which reads the files on your drive and rearranges the clusters so that all files are written into contiguous chains. If you have utility software that does this, running it every six months or so is a good maintenance plan. To

A+ OS 1.3, 1.1, 3.2

graphically see how badly a drive is fragmented, use Norton to view your FAT. Norton highlights the clusters for each file in a different color so you can easily identify all the clusters that belong to a single file. By moving your cursor over the FAT, you can easily see whether your hard drive is badly fragmented.

For Windows 9x, the **Defragmenter** utility is also available. Choose **Start**, then **Programs**, then **Accessories**, and then **System Tools**. The menu in Figure 6-24 is displayed. (Two of the other menu items, ScanDisk and DriveSpace, will be discussed next.) Click **Disk Defragmenter** and select the drive from the dialog box that is displayed. Click OK. When the operation is complete, the message in Figure 6-25 appears; click **Yes** to exit.

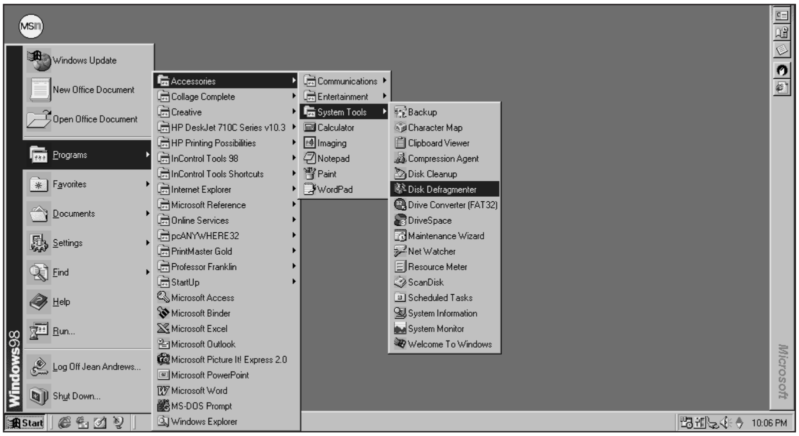

Figure 6-24 Windows 98 utilities

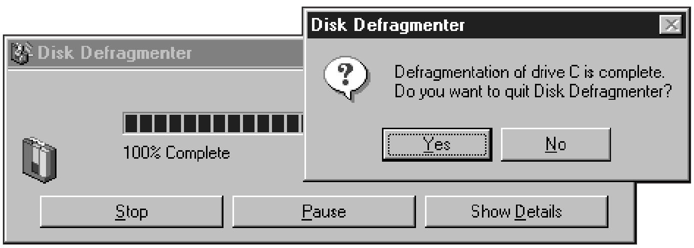

Figure 6-25 Disk Defragmenter results

TIP

Defragmenting a large hard drive may take a long time, so plan for this before you begin.

Cross-Linked and Lost Clusters

As you learned in Chapter 5, the directory on either a floppy disk or hard drive holds the number of the first cluster in the file. The FAT holds the map to all the clusters in a file. Occasionally, the mapping in the FAT becomes corrupted, resulting either in lost clusters or in cross-linked clusters, as shown in Figure 6-26. Here File 3 has lost direction and is pointing to a cluster chain that belongs to File 4. Clusters 29–31 are called **cross-linked clusters** because more than one file points to them, and clusters 15–17 and 28 are called **lost clusters** because no file in the FAT points to them.

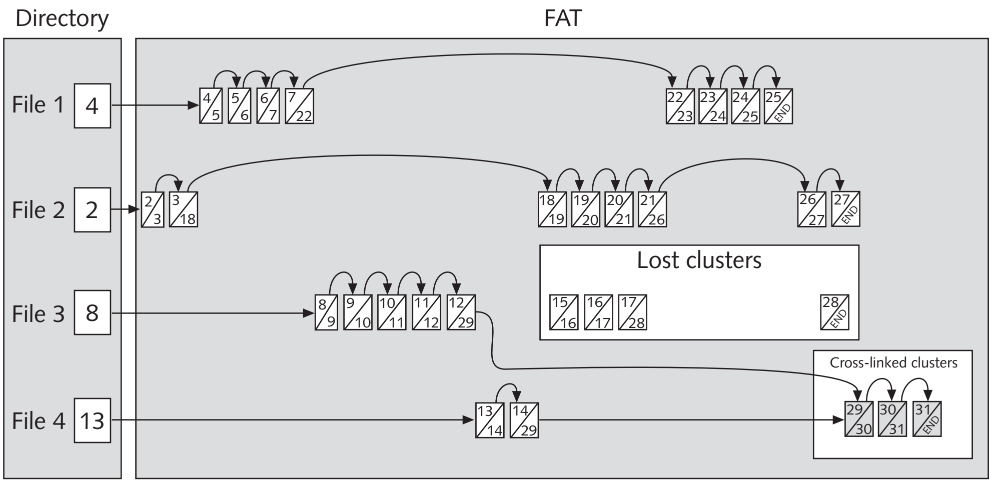

Figure 6-26 Lost and cross-linked clusters

A+ OS 1.3, 1.1, 3.2

To repair cross-linked and lost clusters, use the ScanDisk utility in either DOS or Windows 9x. For DOS, enter the command **SCANDISK** from the DOS prompt. The screen in Figure 6-27 is displayed. When the program finishes scanning the disk, it returns you to a DOS prompt.

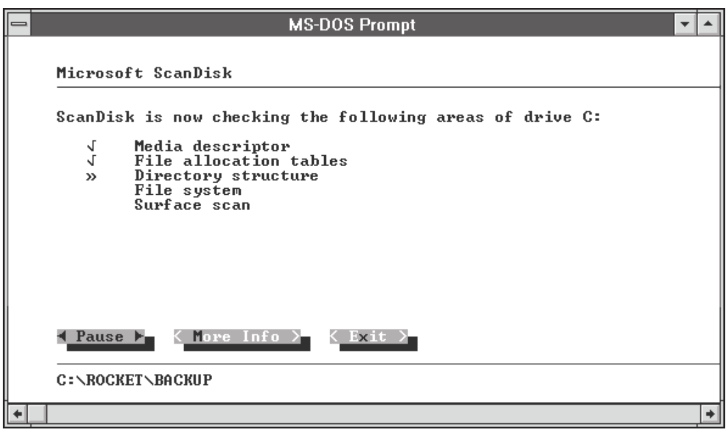

Figure 6-27 SCANDISK for DOS

A+ OS 1.3, 3.2

For Windows 98, click **Start**, then **Programs**, then **Accessories**, then **System Tools**, and then **ScanDisk**, as shown in Figure 6-24. The ScanDisk utility first asks which drive you want to scan and gives you the choice between a standard and thorough scan. The standard scan checks files and folders for errors. The thorough scan does all that the standard scan does plus checks the disk surface. Click **Start** to begin the scan. Errors are reported as in Figure 6-28, and results are displayed as in Figure 6-29.

Figure 6-28 ScanDisk reports errors

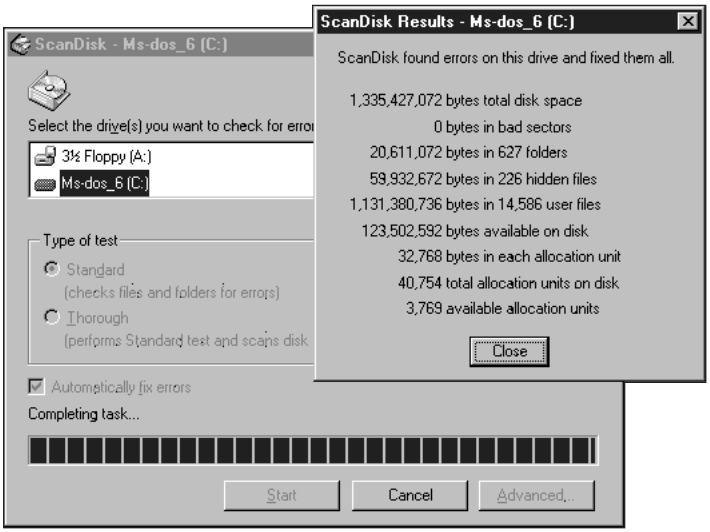

Figure 6-29 ScanDisk results

Disk Compression

Disk compression software can help meet the ever-increasing demands for more space on hard drives to hold improved software. Software packages requiring 100 to 150 MB of hard drive space were unheard of three or four years ago but are now common. The sizes of hard drives have increased proportionately. Even so, we often seek ways to cram more onto nearly full hard drives.

What Is Disk Compression?

Software to manage disk compression works by (1) storing data on your hard drive in one big file and managing the writing of data and programs to that file, and (2) rewriting data in files in a mathematically coded format that uses less space. Most disk compression programs, such as Stacker and DOS DriveSpace, combine these two methods.

Disk Compression in DOS and Windows 3.x

The first method listed above relies on the disk compression software provided by DOS and Windows 3.x, which uses a device driver loaded in the CONFIG.SYS file. The driver treats all the hard drive space as one big file called the host file. When DOS tries to write files to the hard drive, it passes these files to the driver, which does the actual writing. The driver keeps track of where the files are located on the drive. Normally with DOS the smallest allocation unit for a file is 1 cluster, which can be as large as 8 sectors or 4,096 bytes, so that even files physically smaller than one cluster still occupy a full 4,096 bytes. With a disk compression driver, the smallest allocation unit is one sector, or 512 bytes, and sometimes this is even reduced to one-half sector, or 256 bytes. The driver is free to place as many as eight small files into the 4,096-byte cluster space that DOS would use for one small file. This method of disk compression can be very effective if the drive contains many small files. If most files on the drive exceed 4,096 bytes, there may be little gain.

The second method of disk compression for hard drives listed above takes its idea from file compression software like PKZIP. File compression has been around for quite some time. It compresses one or more files into one small file for easy exporting to other systems. The file compression program looks for repeating characters in the data and eliminates the repetition by indicating how many times a character should be written instead of writing the character that many times. The program also writes characters that do not require an entire 8 bits by using only the bits that the character needs. For example, most ASCII characters only use 7 of the 8 bits (see Appendix C). File compression software uses the eighth bit for something else. By making use of every bit and by eliminating repeating characters, this software can compress data to as little as 65% of its original size.

All this sounds very good. You can make an older 500-MB hard drive magically hold 700 MB by adding just one more driver in your CONFIG.SYS file. Caution, however, is in order. Remember that the compression software is putting all your software and data into a single file. Occasionally, one or two files on a hard drive or floppy disk become corrupted. As long as this involves only a single file and rarely happens, you can deal with the loss by recovering the file from a backup, borrowing a fresh copy from a friend, or even reconstructing the file entirely. But with **data compression**, that one file is everything on your hard drive. All your eggs are truly in one basket!

You also have the added complexity of having one more layer of software manipulating your data. If this driver is incompatible with some application you happen to be using one day, the results could be disastrous. Also, because of this added complexity, disk access time is slowed down.

In summary, disk compression does save hard drive space, but you need to carefully consider the risks involved. If you do choose to use disk compression, keep good backups of both the data and the software. If the data and software on your drive are especially valuable, you may want to invest in a larger hard drive instead of using compression.

Disk Compression in Windows 9x

A+ OS 1.3

A **compressed drive** is not a drive at all; it's a file. Figure 6-30 shows the two parts of a compressed drive. The **host drive**, in this case drive H, is not compressed and is usually a very small partition on the drive, generally under 2 MB. The host drive contains a special file called a **CVF** (**compressed volume file**). The CVF holds everything on drive C, compressed into just one file.

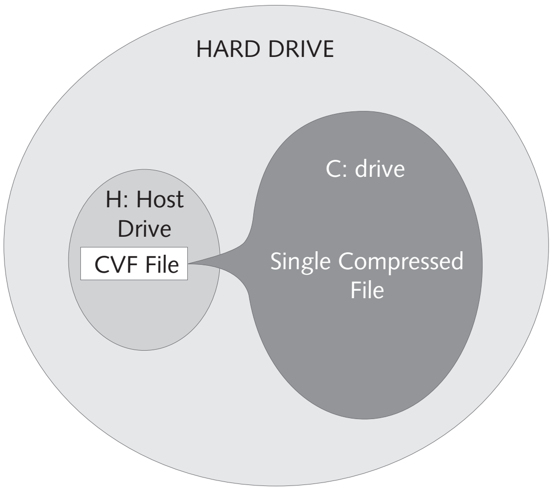

Figure 6-30 A compressed drive

Although there are several disk compression utilities on the market, Windows 9x offers its own, called **DriveSpace**. Others are STAC Electronics Stacker and DoubleSpace, both of which are supported by Windows 9x. DoubleSpace is also available under Windows 3.x. DriveSpace is used in the example here.

TIP

DriveSpace does not work with FAT32 under Windows 98.

DriveSpace does the following things to compress a drive.

- Assigns a different drive letter to the hard drive, such as H
- Compresses the entire contents of the hard drive into a single file on drive H

- Sets up the drive so that Windows 9x and other applications view this compressed file as drive C
- Configures Windows 9x so that each time it boots, the DriveSpace driver will load and manage the compressed drive

Follow these steps to compress a drive in Windows 98 using DriveSpace: Click **Start**, then **Programs**, then **Accessories**, then **System Tools**, as shown in Figure 6-29. Click **DriveSpace** to display the dialog box in Figure 6-31. Drive A is used as the example here, but drive C gives similar results. Select drive A by clicking on it, and then choose **Compress** from the Drive menu. The dialog box in Figure 6-32 is displayed, showing how much space would be created by compressing the drive. Click **Options**, and the Compression Options box in Figure 6-33 appears. Note that you can allow extra space on drive H that will not be compressed. Click **OK** to return to the box in Figure 6-32. Click **Start** to begin compression. A dialog box displays suggesting that you make backups before compressing the drive.

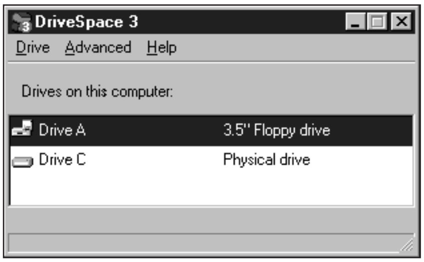

Figure 6-31 Selecting drive using DriveSpace for Windows 98

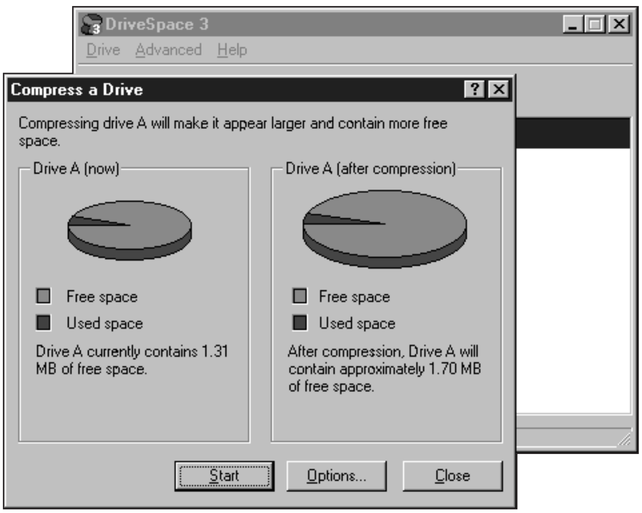

Figure 6-32 Drive compression predictions

6

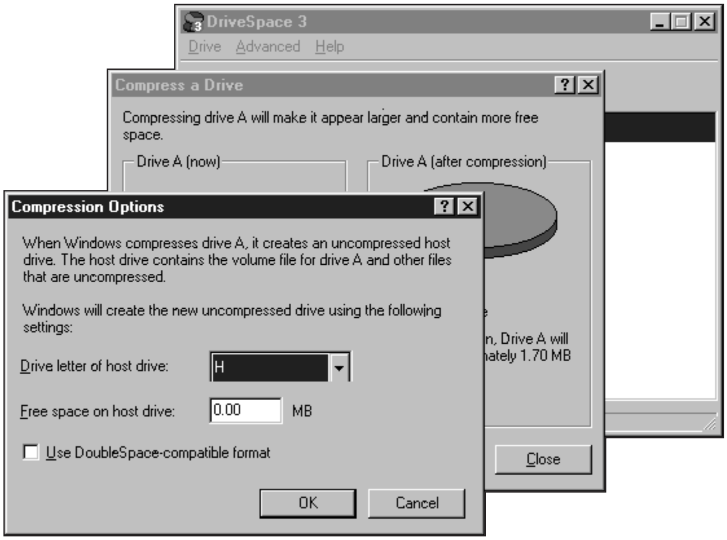

Figure 6-33 Drive compression options

> **TIP** If a Windows 9x utility such as DriveSpace is missing from a menu list, maybe the component was not installed when Windows was installed. To install a Windows component after Windows is installed, go to **Control Panel**, select **Add/Remove Programs**, and click the **Windows Setup** tab.

After a drive is compressed, you can use Explorer to monitor and manage the drive. Right-click the drive letter in Explorer and select **Properties**. Click the **Compression** tab to see information about the compressed drive. Click **Advanced** to see information about the host drive and to run DriveSpace (Figure 6-34).

You can also use Explorer to see how much space compressing a drive will create. Right-click the drive letter and select **Properties,** and then click the **Compression** tab (Figure 6-35). The amount of space made available when a drive is compressed depends on the amount of free space on the drive before compression. To get the most resources out of compression, begin with a relatively free hard drive.

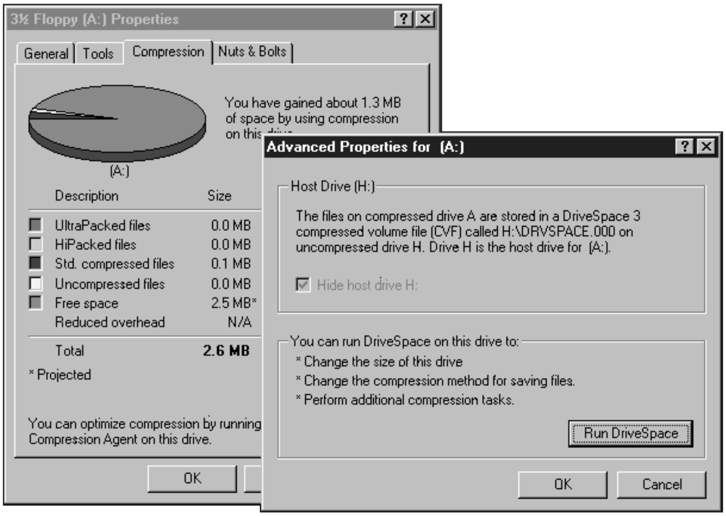

Figure 6-34 The Advanced Properties box gives information about the host drive for a compressed drive

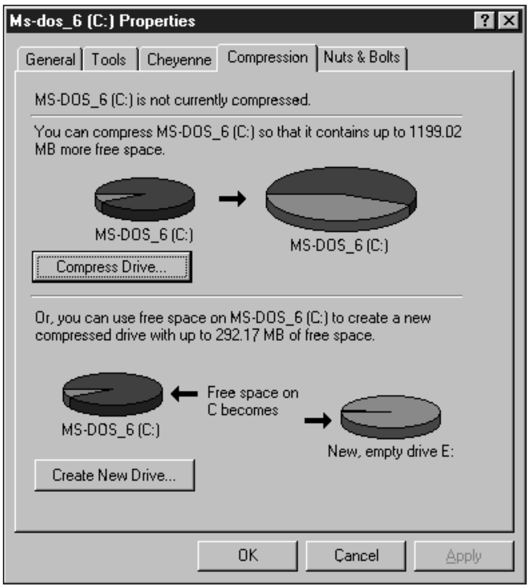

Figure 6-35 The Compression tab on the Properties box gives predictions about compressing a drive

Also notice in Figure 6-35 the option to create a new, empty compressed drive by using the free space on the existing uncompressed drive. Click **Create New Drive**, and Figure 6-36 is displayed. Select the amount of space you want to allot to the new compressed drive and click **Start**. This method is a relatively harmless way to experiment with hard drive compression.

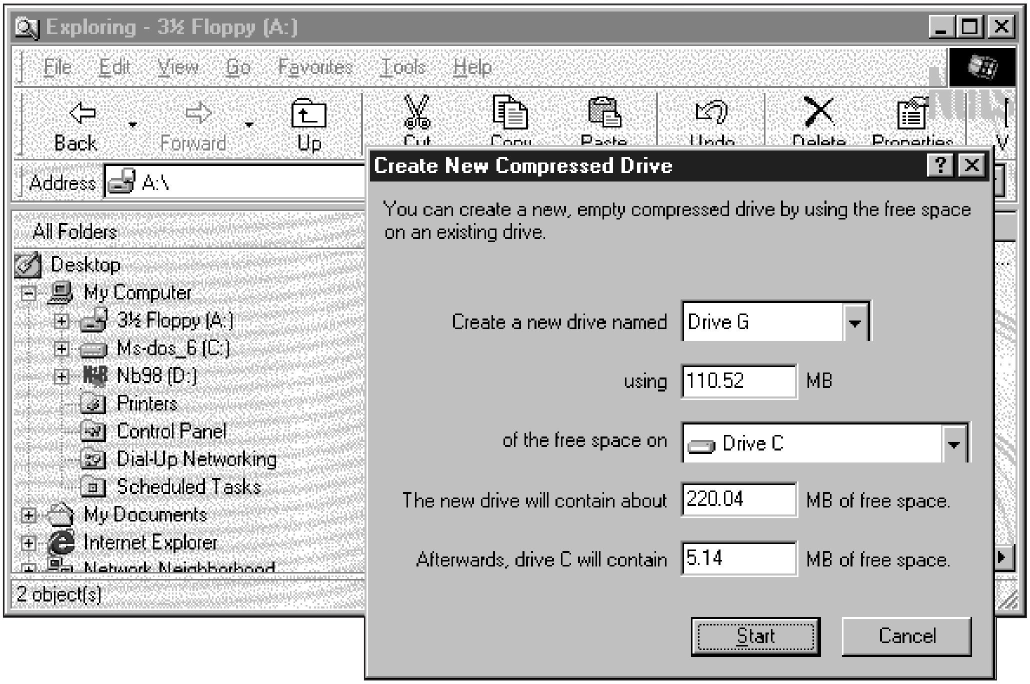

Figure 6-36 A new, empty compressed drive can be created by using free space on an existing drive

A drive can be uncompressed from the DriveSpace Drive menu if you first delete enough files so that the amount of data on the drive does not exceed the uncompressed capacity. If you compress a drive, the driver to manage the compressed drive is loaded only when Windows 9x senses that a compressed drive is present. After the hard drive is uncompressed, Windows 9x no longer automatically loads the mounting driver.

Disk Caching

A **disk cache** is a temporary storage area in RAM for data being read from or written to a hard drive to speed up access time to the drive. The idea behind a cache on a hard drive can be explained as follows:

The CPU asks for data from a hard drive. The hard drive controller sends instructions to the drive to read the data and then sends it to the CPU. The CPU requests more data, quite often data that immediately follows the previously read data on the hard drive. The controller reads the requested data from the drive and sends it to the CPU. Without a cache, each CPU request is handled with a read to the hard drive, as indicated in the top part of Figure 6-37.

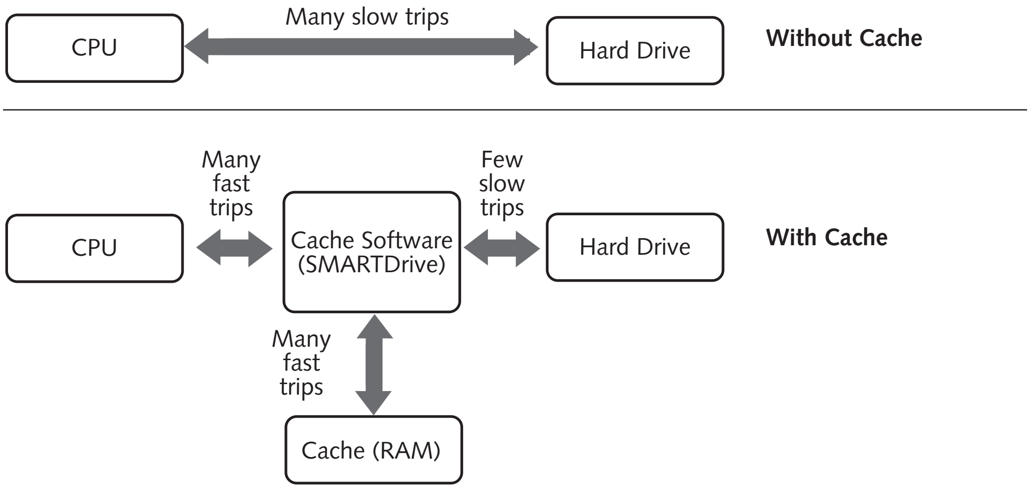

Figure 6-37 A CPU asking a hard drive for data without cache (upper part) and with cache (lower part)

With a hard drive cache, the cache software handles the requests for data, as seen in the lower part of Figure 6-37. The cache program reads ahead of the CPU requests by guessing what data the CPU will request next. Since most data that the CPU requests is in consecutive areas on the drive, the cache program guesses correctly most of the time. The program stores the read-ahead data in memory (RAM). When the CPU requests the next group of data, if the cache program guessed right, the program can send that data to the CPU from memory without having to go back to the hard drive. Some cache software caches entire tracks at a time; others cache groups of sectors.

Hardware Cache or Software Cache

There are two kinds of hard drive caches: a hardware cache and a software cache. Some hard drive controllers have a **hardware cache** built right into the controller circuit board. The BIOS on the controller contains the cache program, and RAM chips on the controller hold the cache.

A **software cache** is a cache program that is stored on the hard drive like other software and is loaded into memory as a TSR, usually when a computer is booted. The software cache program uses system RAM to hold the cache.

There are advantages and disadvantages of each. A hardware cache does not use RAM on the system board, but a software cache uses RAM for both the cache program itself and the data being cached. Therefore, a disadvantage of a software cache is that RAM is used that might otherwise be used for applications software and its data.

On the other hand, a software cache is faster because of where the data is stored. Since data is stored in RAM, when the CPU is ready for the data, the data only needs to travel from RAM to the CPU on the memory bus, the fastest bus on the system board. Since the hardware cache

is on the controller board, when the CPU is ready for data stored in this cache, the data must travel from the controller board over one or more buses to the CPU.

Another disadvantage of a hardware cache is that it is a permanent part of the hard drive controller, and today's hard drives have the controller built into the drive housing. If a faster software cache becomes available, upgrading software stored on your hard drive is a viable option, but exchanging hard drives to upgrade to a faster hardware cache is impractical.

When buying a new hard drive, check whether it includes hardware caches as an option. A controller with its own hardware cache is slightly more expensive than one without a cache.

Disk Cache in DOS and Windows 3.x

SMARTDrive is a 16-bit real mode software cache utility that comes with DOS and Windows 3.x. With DOS 6+, SMARTDrive is executed as a TSR from the AUTOEXEC.BAT file. With earlier versions of DOS, SMARTDrive worked as a device driver and was loaded from the CONFIG.SYS file with the DEVICE= command line. SMARTDrive caches data both being read from and written to the hard drive and caches data being read from floppy disks.

Other popular kinds of cache software for DOS and Windows 3.x are:

- Norton Cache, included in Norton Utilities
- Mace Cache, included in Mace Utilities
- Super PC-Kwik Cache from Multisoft

A+ OS 1.3

VCACHE in Windows 9x

Windows 9x has a built-in 32-bit, protected-mode software cache called **VCACHE**. **VCACHE** is automatically loaded by Windows 9x without entries in CONFIG.SYS or AUTOEXEC.BAT. VCACHE doesn't take up conventional memory or upper memory space the way SMARTDrive does, and it does a much better job of caching. Also, you don't need to tell VCACHE how much memory to allocate to disk caching, as you do with SMARTDrive; VCACHE allocates the amount of memory it uses on the basis of available memory and disk activity.

TIP

In speaking of disk caching, don't confuse a disk cache with a memory cache, discussed in earlier chapters. Memory caching is caching slow memory (DRAM) into fast memory (SRAM). Disk caching is caching slow secondary storage (hard drive) into faster primary storage (RAM).

DOS Buffers

Several years ago, before hard drive caching, buffers were used to speed up disk access. A **buffer** is an area in memory where data waiting to be read or written is temporarily stored. Disk caches do a better job of speeding up disk access than buffers, but many older software packages still in use require that DOS maintain buffers. To specify how many buffers DOS should maintain, use the BUFFERS= command in the CONFIG.SYS file. The only reason

to use buffers today is to satisfy the requirements of older software that uses them. See the software documentation for the recommended number of buffers.

Using DOS under Windows 9x to Manage a Hard Drive

A word of caution: using some DOS commands on a hard drive that uses Windows 9x as the OS may cause damage to a hard drive's file structure. With a Windows 9x upgrade, some of these dangerous commands are erased from the \DOS directory on the hard drive. However, you will find DOS commands that come with Windows 9x stored in the \Windows\Command directory, and some of these should not be used. Here are the ones to avoid:

- Don't use disk utility software that does not know about VFAT, long filenames or FAT32, including older versions of Norton Utilities and Central Point PC Tools.
- Don't use FDISK, FORMAT C:, SYS C:, or CHKDSK while in a DOS session within Windows 9x. However, some of these functions are covered in the next chapter and will be useful to you when you learn what they do and how they do it.
- Don't optimize or defragment your hard drive using software that does not know about long filenames; look for the Windows 95 or Windows 98 compatibility message on the package.
- Don't run hard drive cache programs unless they are written especially for Windows 95 or Windows 98. Remember that Windows 9x has its own built-in caching software.
- Don't use the older DOS backup programs like BACKUP or MSBACKUP, because the long filename information might not be saved during the backup.

Chapter Summary

- Most hard drives today use IDE technology, which has a complex method of organizing tracks and sectors on the disks.
- Older hard drives used either MFM or RLL technology, which used the same number of sectors on every track on the drive.
- The term SCSI hard drive refers more to the bus used by the drive than to the technology of the drive.
- There are several variations of SCSI buses and bus devices, including SCSI-1, SCSI-2, Wide SCSI, Ultra SCSI, and Ultra Wide SCSI.
- Every SCSI bus subsystem requires a host adapter with a SCSI controller and SCSI IDs assigned to each device, including the host adapter.

- Each end of the SCSI bus must have a terminating resistor, which can be either hardware or software.
- Hard drive capacity for drives less than 8.4 GB is determined by the number of heads, tracks, and sectors on the disk, each sector holding 512 bytes of data.
- The operating system views a hard drive through a FAT (file allocation table), which lists clusters, and a directory, which lists files.
- A hard drive is partitioned into logical drives, or volumes. A master boot record at the beginning of the hard drive contains a table of partition information. Each logical drive contains a boot record, FAT, and root directory.
- Physical geometry is the actual organization of heads, tracks, and sectors on the drive, whereas logical geometry is head, track, and sector information that the hard drive controller BIOS presents to system BIOS and the OS. The logical geometry may not be the same as the physical geometry, but, for drives less than 8.4 GB, should yield the same capacity when calculations are made.
- Drives larger than 8.4 GB use LBA mode in which (or "where") the drive is viewed by the BIOS and OS as addressable sectors. The capacity of the drive is calculated as the number of addressable sectors multiplied by 512 bytes per sector.
- System BIOS and software can use CHS, large mode, or LBA mode to manage a hard drive. The size of the drive and the drive manufacturer determine which mode is used.
- The FAT, or file allocation table, lists all clusters on the hard drive and describes how each is allocated. FAT16 uses 16-bit entries and FAT32 uses 32-bit entries to hold the cluster numbers.
- DOS and the first release of Windows 95 support the FAT16 file system for hard drives. Windows 95 Release 2, and Windows 98 support FAT16 and FAT32 file systems. Windows NT supports FAT16 and the NTFS file systems. Windows 2000 supports FAT16, FAT32, and the NTFS file systems.
- The FAT16 file system can be used for drives less than 8.4 GB, but for larger drives, the FAT32 or the NTFS file system must be used.
- Directories on a hard drive hold the information about each file stored on the drive. The main directory created when the drive is first formatted is called the root directory.
- Commands to manage a hard drive include those to create and remove directories, change the attributes on a file, and list paths where the OS can look to find software. DOS and Windows 9x offer commands or menu options to perform these tasks.
- Some ways to optimize drive space and access speed are to reduce fragmentation, to scan the disk for errors, to compress the drive, and to use disk caching.

KEY TERMS

Adapter card — Also called an interface card. A small circuit board inserted in an expansion slot and used to communicate between the system bus and a peripheral device.

Advanced SCSI programming interface (**ASPI**) — A popular device driver that enables operating systems to communicate with a SCSI host adapter. (The "A" originally stood for Adaptec.)

ANSI (**American National Standards Institute**) — A nonprofit organization dedicated to creating trade and communications standards.

ATTRIB command — A DOS command that can display file attributes and even lock files so that they are "read-only" and cannot be modified (for example, ATTRIB +R FILENAME).

Autodetection — A feature of system BIOS that automatically detects the presence of a new hard drive and identifies and configures the drive to CMOS setup.

Batch file — A text file containing a series of DOS instructions to the computer, telling it to perform a specific task (for example, AUTOEXEC.BAT, which contains a series of startup commands).

Block mode — A method of data transfer between hard drive and memory that allows multiple data transfers on a single software interrupt.

Buffer — A temporary memory area where data is kept before being written to a hard drive or sent to a printer, thus reducing the number of writes to the devices.

Burst transfer — A means of sending data across the bus, with one packet immediately following the next, without waiting for clock beats and/or addressing of the information being sent.

CD or CHDIR command — A DOS command to change directories (for example, CD\WINDOWS would change the directory to the Windows directory, and CD\ would return to the Root directory).

Chain — A group of clusters used to hold a single file.

CHS (**cylinders, heads, sectors**) **mode** — The traditional method by which BIOS reads from and writes to hard drives by addressing the correct cylinder, head, and sector. Also called normal mode.

Common access method (**CAM**) — A standard adapter driver used by SCSI.

Compressed drive — A drive whose format has been reorganized in order to store more data. A compressed drive is not a drive at all; it's actually a type of file, typically with a host drive called H.

Cross-linked clusters — Errors caused when files appear to share the same disk space, according to the file allocation table.

CVF (**compressed volume file**) — The file on the host drive of a compressed drive that holds all compressed data.

Data compression — Reducing the size of files by various techniques such as using a shortcut code to represent repeated data.

Defragment — To "optimize" or rewrite a file to a disk in one contiguous chain of clusters, thus speeding up data retrieval.

6

Disk cache — A method whereby recently retrieved data and adjacent data are read into memory in advance, anticipating the next CPU request.

Disk compression — Compressing data on a hard drive to allow more data to be written to the drive.

DriveSpace — A utility that compresses files so that they take up less space on a disk drive, creating a single large file on the disk to hold all the compressed files.

ECHS (**extended CHS**) **mode** —A mode of addressing information on hard drives that range from 504 MB to 8.4 GB, addressing information on a hard drive by translating cylinder, head, and sector information in order to break the 528 MB hard drive barrier. Another name for large mode.

Embedded SCSI device — A SCSI device designed to be a stand-alone SCSI device with some of the host adapter logic built in.

Enhanced BIOS — A newer BIOS that has been written to accommodate larger-capacity gigabyte drives.

Enhanced IDE technology — A newer drive standard that allows systems to recognize drives larger than 504 MB and to handle up to four devices on the same controller.

Folder — A Windows directory for a collection of related files (for instance, a person may find it convenient to create a Mydata directory, or folder, in which to store personal files).

Fragmentation — The distribution of data files, such that they are stored in noncontiguous clusters.

Hardware cache — A disk cache that is contained in RAM chips built right on the disk controller.

Head — The top or bottom surface of one platter on a hard drive. Each platter has two heads.

High-level format — Format performed by the OS that writes a file system to a logical drive. For DOS and Windows 9x, the command used is FORMAT, which writes a FAT and a directory to the drive. Also called OS format.

Host adapter — The circuit board that controls a SCSI bus that supports as many as eight or 16 separate devices, one of which is a host adapter that controls communication with the PC.

Host drive — Typically drive H on a compressed drive. *See* Compressed drive.

Integrated Device Electronics (**IDE**) — A hard drive whose disk controller is integrated into the drive, eliminating the need for a controller cable and thus increasing speed, as well as reducing price.

Large mode — *See* ECHS.

Logical block addressing (**LBA**) — A mode of addressing information on hard drives in which the BIOS and operating system view the drive as one long linear list of LBAs or addressable sectors, permitting drives to be larger than 8.4 GB (LBA 0 is cylinder 0, head 0, and sector 1).

Logical drive — A portion or all of a hard drive partition that is treated by the operating system as though it were a physical drive containing a boot record, FAT, and root directory.

Logical geometry — The number of heads, tracks, and sectors that the BIOS on the hard drive controller presents to the system BIOS and the OS. The logical geometry does not consist of the same values as the physical geometry, although calculations of drive capacity yield the same results.

Logical unit number (**LUN**) — A number from 0 to 15 (also called the SCSI ID) assigned to each SCSI device attached to a daisy chain.

Lost clusters — Lost file fragments that, according to the file allocation table, contain data that does not belong to any file. In DOS, the command CHKDSK/F can free these fragments.

Low-level format — A process (usually performed at the factory) that electronically creates the hard drive cylinders and tests for bad spots on the disk surface.

Master boot record (on a hard drive) — The first sector on a hard drive, which contains the partition table and other information needed by BIOS to access the drive.

MD or MKDIR command — A command used to create a directory on a drive (for example, MD C:\MYDATA).

MIRROR command — An old DOS command that saves information about deleted files as they are deleted. This information can be used later by the UNDELETE command to recover a deleted file. The command can be used to save the partition table to a floppy disk.

Normal mode — *See* CHS.

OS format — *See* high-level format.

Partition — A division of a hard drive that can be used to hold logical drives.

Partition table — A table at the beginning of the hard drive that contains information about each partition on the drive. The partition table is contained in the master boot record.

Physical geometry — The actual layout of heads, tracks, and sectors on a hard drive. *See* Logical geometry.

RD or RMDIR command — A DOS command to remove an unwanted directory (for example, RD C:\OLDDIR). You must delete all files in the directory to be removed, prior to using this command.

Reduced write current — A method whereby less current is used to write data to tracks near the center of the disk, where the bits are closer together.

SCAM (**SCSI configuration automatically**) — A method that follows the Plug and Play standard, to make installations of SCSI devices much easier, assuming that the device is SCAM-compatible.

SCSI (**small computer system interface**) — A faster system-level interface with a host adapter and a bus that can daisy chain as many as seven or 15 other devices.

SCSI bus adapter chip — The chip mounted on the logic board of a hard drive that allows the drive to be a part of a SCSI bus system.

SCSI ID — *See* Logical unit number.

Slack — Wasted space on a hard drive caused by not using all available space at the end of clusters.

SMARTDrive — A hard drive cache program that comes with Windows 3.x and DOS that can be executed as a TSR from the AUTOEXEC.BAT file (for example, DEVICE=SMARTDRV.SYS 2048).

Software cache — Cache controlled by software whereby the cache is stored in RAM.

Terminating resistor — The resistor added at the end of a SCSI chain to dampen the voltage at the end of the chain. *See* Termination.

Termination — A process necessary to prevent an echo effect of power at the end of a SCSI chain resulting in interference with the data transmission. *See* Terminating resistor.

Translation — A technique used by system BIOS and hard drive controller BIOS to break the 504 MB hard drive barrier, whereby a different set of drive parameters are communicated to the OS and other software than that used by the hard drive controller BIOS.

TREE command — A DOS command that shows the disk directories in a graphical layout similar to a family tree (for example, TREE/F shows every filename in all branches of the tree).

UNFORMAT command — A DOS command that performs recovery from an accidental FORMAT, and can repair a damaged partition table if the partition table was previously saved with MIRROR/PARTN.

VCACHE — A built-in Windows 9x 32-bit software cache that doesn't take up conventional memory space or upper memory space, as SmartDrive does.

Virtual file allocation table (VFAT) — A variation of the original DOS 16-bit FAT that allows for long filenames and 32-bit disk access.

Write precompensation — A method whereby data is written faster to the tracks that are near the center of a disk.

Zone bit recording — A method of storing data on a hard drive whereby the drive can have more sectors per track near the outside of the platter.

Review Questions

1. If a hard drive has three platters, how many heads does it have?
2. What two types of tables do DOS and Windows 9x use to manage the data on a hard drive?
3. What is the purpose of the master boot record on a hard drive?
4. Given that there are 512 bytes per sector, calculate the hard drive storage for the following:

   heads: 32, tracks (cylinders): 1024, sectors/track: 63
5. Why does the logical geometry sometimes differ from the physical geometry of a hard drive?
6. How does an OS or other software communicate to the CPU that it wants to access data on a hard drive?

7. What organization is responsible for the IDE/ATA standard?
8. What two factors collectively are responsible for the 504 or 528-MB limit of the IDE/ATA standard?
9. In using large mode to access a hard drive, how is the 504 or 528-MB barrier broken?
10. How can you tell if your PC supports large mode?
11. Which is more popular, LBA mode or large mode? What is the difference between the two?
12. Name four ANSI standards for interfacing to hard drives.
13. How does block mode give faster access to a hard drive? How can you disable block mode?
14. What type of access mode must be used for a 20 GB hard drive?
15. List the steps to report the capacity of a hard drive using Windows 98.
16. What are the advantages of using VFAT?
17. When would it be appropriate to use FAT32 rather than FAT16 on a new hard drive?
18. Give a complete DOS command to create a directory called MYDATA on drive C.
19. Give a complete DOS command to make the file Resume.Doc read-only.
20. List the steps to create a new subfolder under the DATA directory in a Windows 9x system.
21. In Windows 98, how would you examine the file properties of the file Resume.Doc?
22. What causes a file to be fragmented?
23. In Windows 98, what steps would you use to defragment a hard drive?
24. What is the difference between a lost cluster and a cross-linked cluster?
25. Can Windows 98 use DriveSpace to compress a hard drive that is using FAT32?
26. What is one advantage and one disadvantage of a hardware cache over a software cache for a hard drive?
27. SMARTDrive is to Windows 3.1 as ____________ is to Windows 98.
28. What ANSI hard drive interface standard does not use a traditional hard drive cable?
29. What is the difference between narrow SCSI and wide SCSI?
30. When is it not wise to use hard drive compression to increase the capacity of a hard drive?

6

PROJECTS

Using Nuts & Bolts to Manage a Hard Drive

1. Use the Nuts & Bolts Disk Tune utility to defragment your hard drive. Click **Start**, point to **Programs**, **Nuts & Bolts**, and click **Disk Tune**. Select the drive to defragment and click **Next**. When the visual presentation of your hard drive is displayed, answer these questions:
   a. What is the first cluster number that Nuts & Bolts is accessing to defragment?
   b. What is the last cluster number on your hard drive?
   c. Name at least one file on your hard drive that is fragmented.
   d. Name at least one file on your hard drive that is not fragmented.

Examine a Hard Drive's BIOS Settings

1. From the CMOS setup information on your computer, calculate the capacity of the drive. Show your method of calculation.
2. Does your hard drive use LBA mode? If so, compare the capacity of the drive as reported by LBA to the capacity of the drive as reported by ECHS. Are they the same? If they are different, explain why they are different.
3. Write down or print out all the CMOS settings that apply to your hard drive. Explain each setting that you can.

Examine the First Entries at the Beginning of a Hard Drive

(Refer to the optional "Behind the Scenes with DEBUG" sections in Appendix F before tackling this project.)

1. Print out the OS boot record of your hard drive. On the printout, label each item in the record and explain it.
2. Print out the beginning of the first FAT on your hard drive (see Figure F-7). (*Hint*: Use DEBUG and Print Screen.)
3. Calculate the memory location of the beginning of the second copy of the FAT. Show your method of calculation.
4. Calculate the memory location of the beginning of the root directory. Show your method of calculation.
5. Get a printout of the beginning of the root directory of your hard drive. Identify the first five entries in the root directory and the number of clusters in each entry that make up a file. For each of the five entries, explain the meaning of each ON bit in the attribute byte.

Using DOS to Manage a Hard Drive

1. Print the contents of SCANDISK.INI in the \DOS directory of your hard drive. SCANDISK.INI contains settings that SCANDISK for DOS reads and uses when it executes. The Windows 95 version of ScanDisk does not use these settings.
2. What is the setting that allows SCANDISK to delete the contents of a lost cluster without prompting you first?
3. From the INI file information, can SCANDISK repair a damaged boot sector of a compressed drive?
4. Run **SCANDISK** from a DOS prompt. What errors did it find?
5. From a DOS prompt, type **HELP DEFRAG**. What are the cautions listed concerning when not to use DEFRAG?
6. Following the directions from the Help utility, defragment a disk or hard drive using the DEFRAG utility.
7. From a DOS prompt, type **HELP SMARTDRV**, and then print the help information about the SMARTDrive utility.
8. Look in either the CONFIG.SYS or AUTOEXEC.BAT file of your computer for the SMARTDrive command. It should be in one or the other of the two files, but not both. Using your printout, explain each option used in the command line.

Using Windows 9x to Manage a Hard Drive

1. Using the chapter example of Windows 9x DriveSpace, practice disk compression by compressing two disks. Use one newly formatted disk and one disk about half full of data. Compare the results of the two compressions.
2. With your instructor's permission, use the Defragmenter utility to defragment the hard drive of your computer. Don't use this utility if your disk has been compressed by a utility program other than Windows 9x. If you haven't used Windows 9x to compress the drive, look at the documentation for your compression software to see if it offers a defragmenting utility, and use that to defragment the drive.
3. If you are not using add-on utility software to compress your hard drive, use Windows 9x ScanDisk to repair any cross-linked or lost clusters on the drive. If you are using utility software other than Windows 9x to compress your hard drive, use that software utility to scan for cross-linked or lost clusters.
4. Create a file on an empty disk with a long filename. Using DEBUG, display the root directory entries for this file. What are the two filenames in the root directory?

Recover a File

Using a word processor, create and save a short document under the name Test.del. Using Windows Explorer, delete the newly created document. Open the **Recycle Bin**, and select **File**, **Restore** to undelete the file Test.del.

Practicing DOS

Perform the following procedures and commands. For a dot matrix printer, adjust the paper in the printer to start exactly at the top of the page. If possible, attempt to complete all steps on one page only. Use the section on "Using DOS to Manage a Hard Drive" as a reference.

1. Cold boot the computer. Go to the DOS prompt.
2. List the directory of drive C in wide format.
3. FORMAT a new system disk in drive A. (Use your last name for the volume label.)
4. Switch to drive A. (Look for the A:\> prompt.) List the directory of the newly formatted disk.
5. Switch back to drive C. (Look for the C:\> prompt.)
6. Use the COPY command and the wildcard (*) to copy all files in the root directory to your disk in drive A.
7. Display the current DATE on the monitor.
8. Perform a SCANDISK of drive A.
9. Switch to Drive A. (Be sure the A:\> prompt appears.)
10. ERASE all .COM files from the disk in drive A.
11. List the directory of drive A.
12. Print the screen contents. Take the printer offline (press the Online button to make the Online light go off). Use the Line Feed or Form Feed button to eject the rest of the page from the printer.